SCIENTIFIC PROGRAMME

INVITED SPEAKERS

A Case for on-line data analysis for large-scale scientific simulations

Professor Alok Choudhary
ECE Department and Kellogg School of Management
Northwestern University

Abstract

Large-scale scientific simulation has emerged as an attractive mechanism and methodology for understanding nature, for predictions, for designing new materials, Chips, drugs, aircrafts and thousands of other applications. Simulations have accelerated understanding and validating theories and have emerged as alternatives to expensive and time consuming experiments. However, as simulations scale to really large sizes and systems, it is becoming impractical to analyze results of simulations manually or based on pure visualization.

Simulations can generate hundreds of GBs to 100's of TBs of data for analysis. How fast and cost effectively that analysis can be performed is crucial for knowledge discovery, steering of new models and simulations as well as experiments. Thus manual process for this analysis is a bottleneck. In fact, this has resulted in a large-scale data management and analysis problem. scientific experiments and simulations is a challenging tasks. Current data management and analysis techniques do not measure up to the challenges posed by such requirements in terms of performance, scalability, ease of use and interfaces. Tera-scale computing requires newer models and approaches to solving the problems in storing, retrieving, managing, sharing, visualizing, organizing and analyzing data at such a massive scale.

In this talk I will describe some problems and challenges for large-scale data management and analysis for scientific discoveries for data generated from simulations. I will make a case for on-line analysis and data mining and scalable data management as tools for accelerating knowledge discoveries. A system to illustrate the ideas as well as recent results validating this approach will be presented.

Biography

Alok Choudhary is currently professor in the Electrical and Computer Engineering Department and The Kellogg School of Management at Northwestern University since September. From 1989 to 1996 he was an a faculty in the ECE department at Syracuse University. Alok Choudhary received his Ph.D. from University of Illinois, Urbana-Champaign, in Electrical and Computer Engineering, in 1989 and an M.S. from University of Massachusetts, Amherst, in 1986.

He received the National Science Foundation's Young Investigator Award in 1993, an IEEE Engineering Foundation award and an IBM Faculty Development award. His main research interests are in high-performance computing and communication systems and their applications in many domains including information processing (e.g., data mining) and scientific computing (e.g., scientific discoveries). In particular, his interests lie in the design and evaluation of architectures and software systems (from system software such as runtime systems to compilers), high-performance servers, high-performance databases and input-output. He has published more than 250 papers in various journals and conferences in the above areas. He has also written a book and several book chapters on the above topics.

DEVELOPMENT AND EXECUTION OF HPC APPLICATIONS ON CLUSTERS AND GRIDS BY P-GRADE

Peter Kacsuk
MTA SZTAKI and Univ. of Westminster
MTA SZTAKI, Lab. of Parallel and Distributed Systems
H-1111 Budapest, Kende u. 13, Hungary
email: kacsuk@sztaki.hu

ABSTRACT

The paper describes the latest novel features of the P-GRADE (Parallel Grid Run-time and Application Development Environment) system. P-GRADE enables the usage of the same high-level graphical environment to develop parallel programs for supercomputers, clusters and the Grid, as well as to run the developed parallel program on those systems. Integrating P-GRADE with Condor flocking was the first step to use P-GRADE for the Grid. The parallel check-point mechanism introduced in P-GRADE guarantees the same fault-tolerance for PVM jobs as the standard universe guarantees for sequential jobs in Condor. Parallel programs developed by P-GRADE can run both in Condor and Globus based Grids either as PVM or MPI programs. The workflow layer of P-GRADE supports component based application development and enables the solution of very complex problems in the Grid.

1 INTRODUCTION

Grid computing has been originated from metacomputing where the main goal was to execute grand-challenge applications at several supercomputers simultaneously in order to reduce the actual execution time. In the recent years Grid computing was strongly shifted towards a much more general, service-oriented direction where metacomputing is just one possible aspects of the many others aspects of the Grid. Nevertheless, metacomputing is still an important branch of Grid computing and programming environments able to support this branch are extremely important.

Such a programming environment is P-GRADE (Parallel Grid Run-time and Application Development Environment) that has the following man features:

1. Supports the whole life-cycle of parallel program development.
2. Provides a complete solution for efficient and easy parallel program development for non-specialist programmers (like chemists, biologists, etc.).
3. Supports fast parallelization of sequential programs.
4. Provides a unified graphical support in program design, editing, debugging and performance analysis.
5. Supports the message-passing parallel programming paradigm by its hybrid (partially graphical, partially textual) language, called GRAPNEL.
6. P-GRADE generates from a GRAPNEL program either PVM or MPI code according to the user's need.
7. Its run-time system is highly portable: a parallel program developed under P-GRADE can run nearly transparently on supercomputers, clusters and in the Grid.
8. Currently P-GRADE is integrated with Condor (Thain et al. 2003) and Globus (Foster and Kesselman 1999), and hence it can be used to run PVM and MPI jobs in any Grid system which is based on Condor and/or Globus.
9. The P-GRADE run-time system is extended with automatic parallel program check-point and hence either processes of a parallel application or complete applications can migrate in the Grid. (Condor can provide this feature only for sequential jobs).
10. The MPICH-G2 code generator of P-GRADE and the Globus integration enables P-GRADE to run multi-site MPI jobs in any Grid system which is based on Globus.
11. The workflow layer of P-GRADE supports component-based application development and generates Condor DAGMan workflow definition. Workflows generated by P-GRADE are executable in any Globus and Condor based Grid systems.

In short, P-GRADE provides a powerful programming environment both for Condor and Globus based Grid systems. The paper describes those features of P-GRADE that are used for the Condor-based Grid systems. Section 2 describes the interactive mode of P-GRADE. Section 3 defines the job mode of P-GRADE showing its integration with Condor. Section 4 shows how the P-GRADE workflow system can be used in Condor and Globus based Grid systems.

2 P-GRADE

In order to cope with the extra complexity of parallel and distributed programs arising due to inter-process communication and synchronization, we have designed a graphical programming environment called P-GRADE. Its major goal is to provide an easy-to-use, integrated set of programming tools for development of general message-

passing applications to be run on both homogeneous and heterogeneous distributed computing systems like supercomputers, clusters and Grid systems. The central idea of P-GRADE is to support each stage of the parallel program development life-cycle by an integrated graphical environment.

The first stage of the life-cycle is the program design which is supported by the GRAPNEL (GRAphical Process NEt Language) language and the GRED graphical editor. In GRAPNEL, all process management and inter-process communication activities are defined graphically in the user's application. Low-level details of the underlying message-passing system are hidden. P-GRADE generates all message-passing library calls automatically on the basis of the graphical code of GRAPNEL. Since graphics hides all the low level details of message-passing, P-GRADE is an ideal programming environment for application programmers who are not experienced in parallel programming (e.g., for chemists, biologists, etc.). GRAPNEL is a hybrid language: while graphics is introduced to define parallel activities, textual language parts (C/C++, FORTRAN or Java) are used to describe sequential activities.

GRAPNEL is based on a hierarchical design concept supporting both the bottom-up and top-down design approaches. A GRAPNEL program has three hierarchical layers which are as follows from top to bottom:

- Application layer is a graphical layer which is used to define the component processes, their communication ports as well as their connecting communication channels. Shortly, the Application layer serves for describing the interconnection topology of the component processes. An example is shown in Fig. 1.
- Process layer is also a graphical layer to define the internal structure of the component processes by a flow-chart like graph (see Fig. 1). The basic goal is to provide graphical representation for the message passing function calls. As a consequence, every structure that contains message passing calls should be graphically represented. The following types of graphical blocks are applied: loop construct, conditional construct, sequential block, message passing activity block and graph block. Sequential blocks must not contain any message passing calls.
- Text layer is used to define those parts of the program that are inherently sequential and hence a textual language like C/C++, Java or FORTRAN can be applied at this level. These textual codes are defined inside the sequential blocks of the Process layer (see Fig. 1).

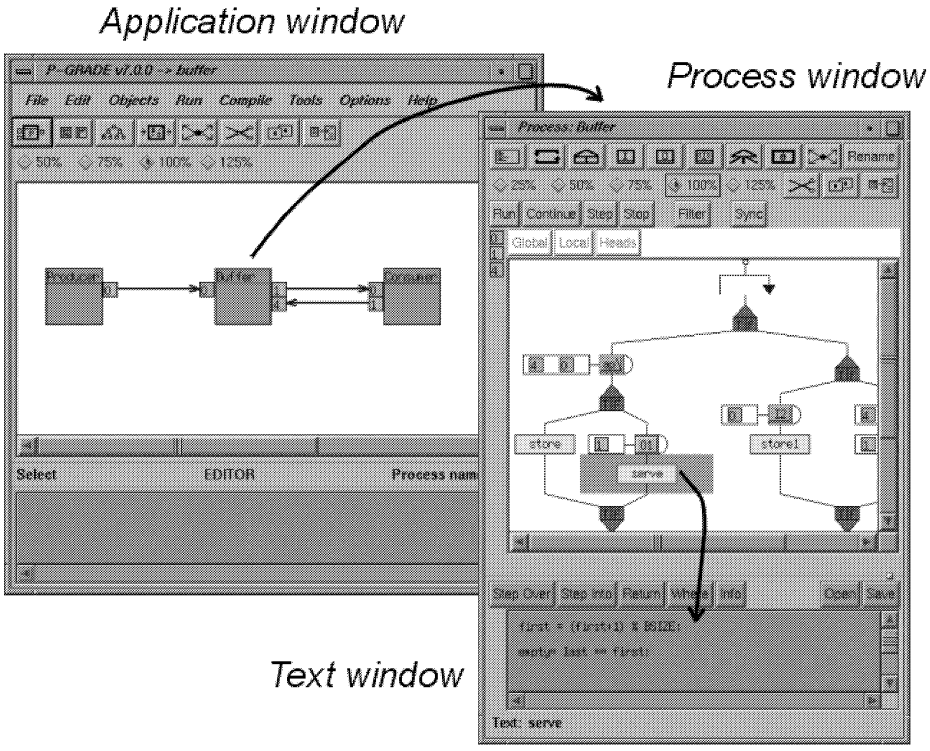


Fig. 1: Hierarchical layers of GRAPNEL programs and the supporting window types

The top-down design method can be used to describe parallel activities of the application program. At the top level the topology and protocols of the interprocess communication can be defined and then in the next layer the internal structure of individual processes can be specified. At this level and in the Text layer the bottom-up and top-down design methods can be used in a mixed way. In the case of the top-down design method, the user can define the graphical structure of the process and then uses the Text layer to define the C/C++ or FORTRAN code for the sequential blocks. In the bottom-up design approach, the user can inherit code from existing C/C++ or FORTRAN libraries and then can build up the internal process structure based on these inherited functions. Moreover, GRAPNEL provides predefined scalable process communication templates that allow the user generate large process farm, pipeline and mesh applications fast and safely. (Process farms are shown in Fig. 3.)

The GRED editor helps the user to construct the graphical parts of GRAPNEL programs in an efficient and fast way. GRAPNEL programs edited by GRED are saved into an internal file called GRP file that contains both the graphical and textual information of GRAPNEL programs. The main concepts of GRAPNEL and GRED are described in detail in (Kacsuk 2000a).

The second stage is the pre-compilation and compilation of GRAPNEL programs. The goal of pre-compilation is to translate the graphical language information of the GRP file into PVM or MPI function calls and to generate the C or FORTRAN source code of the GRAPNEL program. For the sake of flexibility, PVM and MPI function calls are not called directly in the resulting C code, they are hidden in an internal library, called the GRAPNEL Library which has two versions. In the first version (GRP-PVM Library) the GRAPNEL Library functions are realized by PVM calls, and in the second version (GRP-MPI Library) they are realized by MPI function calls. For compiling and linking, any standard C compiler and linker can be used. The linker uses the following libraries:

- PVM, GRP-PVM (in the case of PVM communication system), GRM
- MPI, GRP-MPI (in the case of MPI communication system), GRM

The GRM monitoring library is optional, it is needed only if performance monitoring is applied at run time. Details of the code generator are described in (Drótos et al. 2001).

The third stage is mapping in order to allocate processes to processors. Mapping can be done by a very simple mapping table generated by P-GRADE. The mapping table can be easily modified by simple mouse clicks. The mapping information is also inserted into the GRP file.

Having the necessary executables for the parallel/distributed computing system, the next stage is validating and debugging the code. The DIWIDE distributed debugger has been developed for P-GRADE in which the debugging information is related directly back to the user's graphical code. DIWIDE applies a novel macrostep debugging approach (Kacsuk 2000b) based on which both replay and systematic debugging is possible and DIWIDE automatically detects dead-lock in the message-passing code. GRAPNEL programs can be executed step-by-step both at the usual C instruction level, at the higher level graphical icon level and at the macrostep level. These features significantly facilitate parallel debugging which is the most time-consuming stage of parallel program development.

After debugging the code, the next step is performance analysis. First, it requires performance monitoring which generates an event trace file at program execution time and then, performance visualization which displays performance oriented information by several graphical views. Performance monitoring is performed by the P-GRADE Monitor (GRM) that can support both PVM and MPI (Balaton et al. 2001). PROVE, the performance visualization tool of P-GRADE, can provide both on-line and off-line visualization based on the output trace file of GRM (Kacsuk 2000c). (See an example on Fig. 4.)

3 INTEGRATING P-GRADE AND CONDOR

The interactive working mode of P-GRADE can be used in supercomputers and clusters when those resources are dedicated for a particular application. However, Grid resources are typically not dedicated for a single Grid application and hence Grid resources are exploited by jobs that can be located and controlled by Grid job managers. As a consequence since P-GRADE is aimed at supporting the development and control of parallel Grid applications, too it should support the Grid job execution mode. In order to achieve this P-GRADE was integrated with Condor.

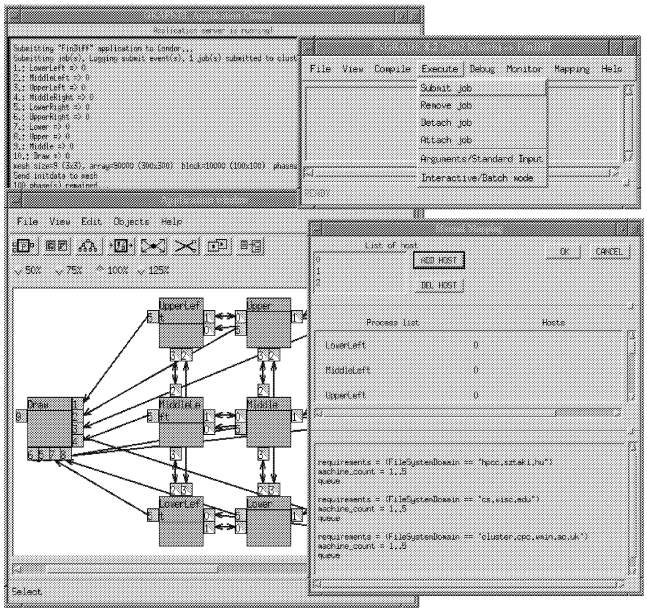


Fig. 2. Snapshot of the P-GRADE/Condor system

Condor is a resource manager that can support high-throughput computing in clusters and in the Grid. The most advanced feature of Condor is the classads mechanism by which it can match application programs with execution resources. When a user submits a job she has to describe the resource requirements and preferences of her job. Similarly, resource providers should advertise their resources by configuration files. The Condor Matchmaker process tries to match jobs and resources by matching the resource requirements and resources configuration files. When such a matching occurs the Matchmaker process notifies both the submitter machine and the selected resource. Then, the submitter machine can send the job to the selected machine that will act as an execution machine.

Integration of P-GRADE and Condor means that after developing a parallel program in the interactive mode the P-GRADE user can switch to batch-mode inside P-GRADE and in such case program execution will result in the automatic generation of a parallel Condor job. P-GRADE will automatically construct the necessary Condor job description file containing the resource requirements of the parallel job. The mapping function of P-GRADE is changed according to the Condor needs. In Condor the user can define machine classes from which Condor can reserve as many machines as it is defined by the user in the job description file.

When the user generates a Condor job under P-GRADE, creation of the Condor mapping is the only Condor-related task. P-GRADE supports this activity by offering a default set of machine classes for the user in the mapping phase. After submitting the Condor job under P-GRADE the user can detach P-GRADE from the job. It means that the job does not need the supervision of P-GRADE when it is executed in the Grid by Condor. Meanwhile the P-GRADE generated Condor job is running in the Grid, P-GRADE can be turned off or it can be used for developing other parallel applications. However, at any time the user can attach again the job to P-GRADE in order to observe the current status and results of the job inside the P-GRADE environment.

The integration of P-GRADE and Condor was a joint work of the P-GRADE team at SZTAKI and the Condor team at the Univ. of Wisconsin-Madison. The integrated system was presented at the CCGrid'2002 conference where a finite difference parallel application (FinDiff) shown in Figure 4 was launched from Berlin and it was simultaneously running on three different clusters (at Univ. of Wisconsin-Madison, SZTAKI and Univ. of Westminster) using the flocking techniques of Condor. The Application Window in Fig. 2 shows the inter-process topology of the parallel application. The Manual Mapping window (bottom right) was used to allocate the processes of the application to different clusters. The clusters are represented as Condor classes (see the lower part of the Manual Mapping window) and identified by 0, 1 and 2 shown in the upper part of the mapping window. Finally, the middle part is used to allocate processes of the application to the available resource classes.

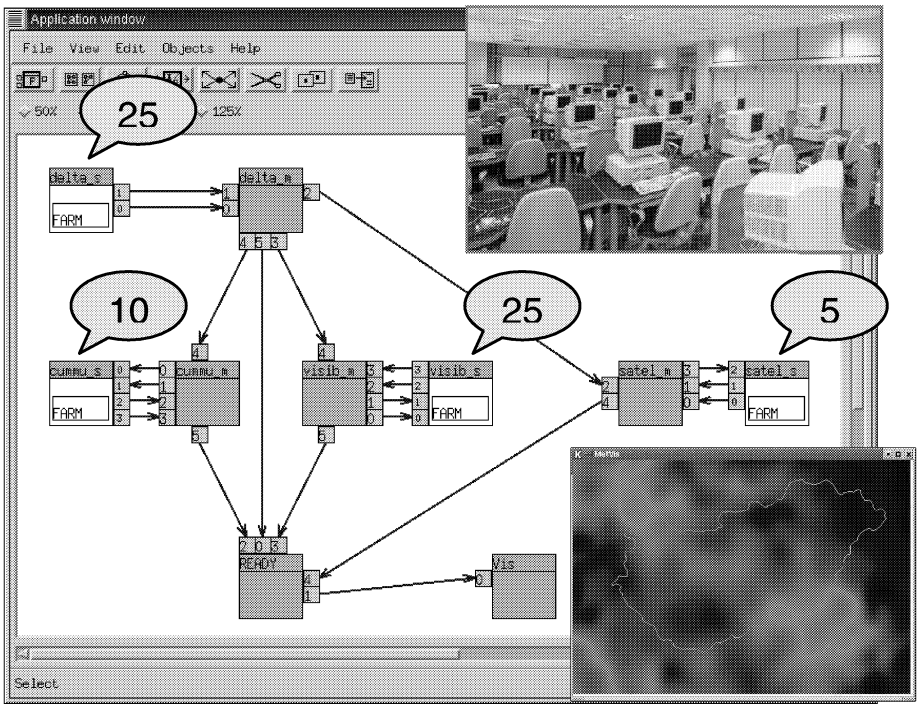


Fig. 3. Parallel version of the MEANDER program and its output

A great advantage of using Condor in the Grid is that Condor contains a checkpoint mechanism for sequential jobs. Based on this checkpoint facility Condor can provide fault-tolerant operation for sequential jobs, i.e., it can guarantee that sooner or later every sequential Condor job will be finished in the Grid. However, Condor cannot create checkpoints for generic parallel programs. In the integrated P-GRADE/Condor system we have solved this problem. If the P-GRADE environment is requested by the user to create a parallel PVM job for Condor, such PVM program can be checkpointed by the integrated P-GRADE/Condor environment and hence its completion can be guaranteed even in a dynamically changing environment like the Grid.

The job mode of P-GRADE was demonstrated by the MEANDER nowcast program package of the Hungarian Meteorology Service at the EuroPar'2003 conference in Klagenfurt. The goal of the MEANDER package is to provide ultra-short range (up to 1 hour) weather forecast of dangerous situations like storm and fog on a high resolution regular mesh (10km → 1km). To achieve this goal we have parallelized six out of the ten component algorithms of MEANDER by P-GRADE (Lovas et al. 2002).

At Klagenfurt, we used a simplified version of MEANDER containing only 4 algorithms for the demonstration. The P-GRADE view of this MEANDER version is shown in Fig. 3. where the four algorithms are realized by the processor farm concept. The numbers in the clouds represent the number of processors that were used in the farms during the demonstration. First the delta algorithm should be executed, then in the second phase, the other three algorithms can be executed in parallel. At that time 40 processors were used in parallel. Finally, a visualization process is applied to draw the weather map shown in the right bottom of Fig. 3.

One parallel job was generated by P-GRADE and passed to Condor at Klagenfurt. Condor selected the 58 processor Linux cluster of SZTAKI (shown in the right top of Fig. 3) to execute the job. Then P-GRADE transferred the job to the SZTAKI cluster, collected the necessary meteorology database input file from the Hungarian Meteorology Service and passed the job with all the files to Condor at the SZTAKI cluster where Condor took care of the parallel execution of the job. Finally, when the whole job was finished, P-GRADE took care of transferring the result file back to Klagenfurt and removing the temporary directory it created for the job at the SZTAKI cluster.

Another important advantage of the integrated Condor/P-GRADE system is that it enables the on-line application monitoring for the parallel Condor jobs. Notice that in the original Condor system such application monitoring is not possible. Only the status of the job can be monitored under Condor but the internal behaviour of processes of the parallel job is invisible for the user. In case of the integrated Condor/P-GRADE system the GRM/PROVE tools support the same kind of on-line observation that is possible on supercomputers and clusters in the interactive mode. In order to support the on-line monitoring and visualization of parallel Grid applications we have developed a Grid Application Monitoring Infrastructure (GAMI) as part of the EU DataGrid and GridLab projects. The upper layer of GAMI contains the GRM/PROVE tools while the lower layer is realized by the Mercury Grid monitor (Balaton and Gombás 2003).

We applied the GAMI infrastructure during the demonstration in Klagenfurt. A Mercury monitor daemon was started at every Grid site. When monitoring was requested, the parallel program inside the Condor job was connecting to the local monitor of Mercury, that sent back the collected trace file to the submit machine at Klagenfurt. Whenever the user requested a trace collection by PROVE it visualized the current status of the parallel program execution on the selected Grid site. The final trace file visualization picture is shown in Fig. 4. The picture clearly shows that first the delta algorithm was running on 25 processors and when it was finished, it triggered the execution of the other three algorithms that were executed simultaneously.

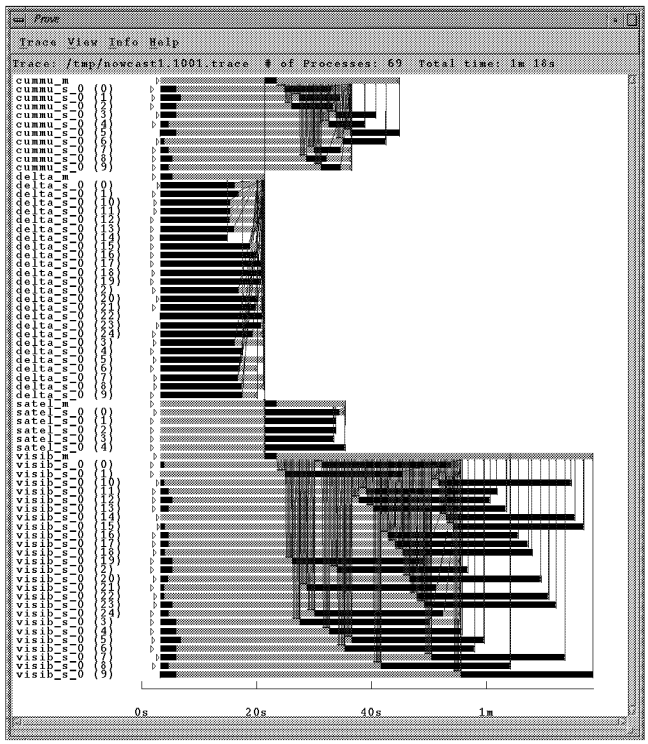


Fig. 4. Execution visualization of the MEANDER program

4 P-GRADE WORKFLOW LAYER

A new extension of P-GRADE is the workflow support that enables constructing and executing very complex applications in the Grid. The workflow version of the MEANDER program depicted in Fig. 5. The structure of the graphs in the Workflow Window and in the Application

Window of P-GRADE are graphically very similar but the meaning of the graph components are quite different. The nodes of the Application Window graph are interpreted as processes and their communication (represented by the ports and arcs of the graph) is interpreted as message passing communication which is implemented either by PVM or MPI. In the workflow graph, the nodes are jobs, the ports represent files (green ports are input files, grey ports are output files) and the arcs denote file transfer between two jobs. For example job "satel" in Figure 5. requires one input file (from job "delta") and produces one output file that is sent to job "metvis".

The workflow layer supports the component based creation of large applications. The nodes of the workflow can be one of the following four component types:

- sequential component
- PVM component
- MPI component
- GRAPNEL component

For example, jobs "delta", "cummu", "visib" and "satel" are GRAPNEL components (notation used: GRP) and job "metvis" is a sequential one (notation used: SEQ). This distinction is necessary because the job manager on the selected Grid site should be able to support the corresponding parallel execution mode (see various Condor universes). A GRAPNEL component is translated into a PVM or MPI job but it should be distinguished from the other types of parallel components since P-GRADE provides full interactive development support for GRAPNEL components. The executables of the components can be used either from existing libraries or can be developed by P-GRADE. In case of sequential, PVM and MPI components the executables should be taken from existing libraries. In case of the GRAPNEL components the interactive mode of P-GRADE can be used for developing and testing the parallel code. By simply clicking on such a node of the workflow graph, P-GRADE opens the Application Window in which the parallel program of the GRAPNEL component can be defined and modified. It means that the workflow layer represents a new P-GRADE layer additionally on top of the already showed three layers.

Besides the type of the components, the user must specify the name of the executable, the necessary arguments and the execution resource requirements (architecture, operating system, minimal memory and disk size, number of processors, etc.) for each job. Obviously, for parallel jobs supercomputers or clusters should be specified as resource requirements and it is the task of the underlying Grid Application Manager to select the appropriate Grid site for such parallel job. To specify the resource requirements, P-GRADE uses the Condor resource specification syntax and semantics.

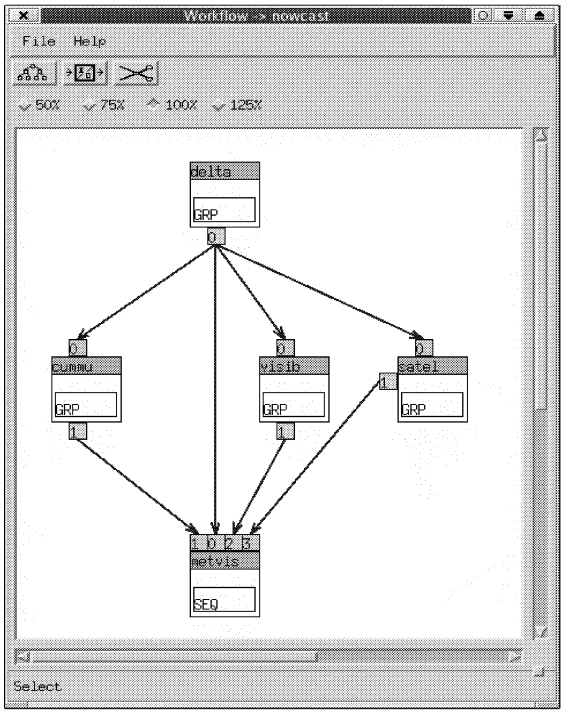


Fig. 5. The workflow version of the MEANDER program

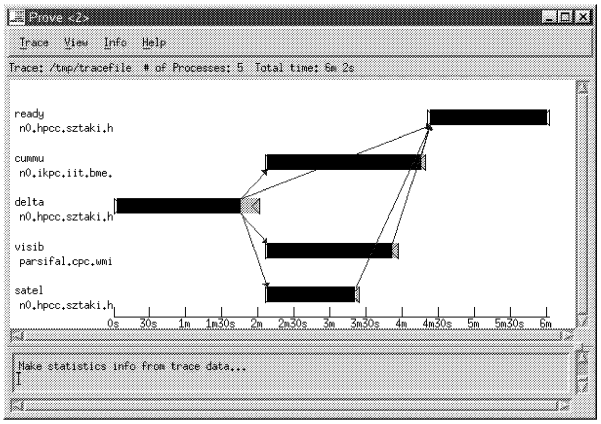


Fig. 6. Execution visualization of the workflow version of the MEANDER program

Concerning the execution of the workflow, it is a generalization of the Condor job mode described in Section 2. To execute the workflow we use the Condor DAGman tool by generating

- a Condor submit file for each node of the workflow graph (already available from the Condor job mode)
- a DAGman input file that contains the following information:

1. List of jobs of the workflow (associating the jobs with their submit files)
2. Execution order of jobs in textual form as relations
3. The number of re-executions for each job's abort
4. Tasks to be executed before starting a job and after finishing the job (The PRE and POST scripts generated from the P-GRADE workflow realize the necessary input and output file transfer operations among the jobs of the workflow.).

The actual execution of the workflow is automatically started from P-GRADE by calling the

condor_submit_dag

command with the DAGman input file described above. If the P-GRADE system is running in a Condor pool, the command is immediately interpreted and executed by Condor. If the P-GRADE submit machine is not in a Condor pool, the following extra operations are supported by P-GRADE:

1. A remote Condor pool should be selected by the user via the Mapping Window of P-GRADE.
2. All the necessary files (executables, input files, DAGman input file, Condor submit files) are transferred to a machine of the selected Condor pool.
3. The "condor_submit_dag" command is called in the selected Condor pool.
4. After finishing the workflow execution, the necessary files are transferred back to the P-GRADE client machine.

The execution of the workflow can be monitored and visualized by the same GAMI infrastructure that is applied for the individual Condor jobs. Fig. 6 shows the execution visualization of the MEANDER workflow on three clusters (SZTAKI, Univ. of Westminster and Technical Univ. of Budapest).

Notice that currently we use Condor DAGman as the workflow engine. However, in the near future we are going to create a general Grid Application Manager that takes care of all possible optimizations concerning the selection of computing sites and file resources in the Grid, controlling the migration of jobs of the workflow among different Grid resources, etc. The Grid Application Manager will allow to run a workflow application across several different Grids provided that those Grids have a gateway mechanism to communicate.

5 CONCLUSIONS

P-GRADE provides a high-level graphical environment to develop parallel applications transparently both for parallel systems and the Grid. One of the main advantages of P-GRADE is that the user has not to learn the different APIs for parallel systems and the Grid, simply by using the same environment will result in a parallel application transparently applicable either for supercomputers, clusters or the Grid. The current version of P-GRADE supports the interactive execution of parallel programs as well as the creation of a Condor job to execute the parallel program in the Grid. That Condor job can be executed in ClusterGrids where the Condor flocking mechanism is applied as the job transfer mechanism, or in generic Globus based Grids where Condor-G and the GRAM of Condor realize the job transfer mechanism among the Grid sites. The integrated P-GRADE/Condor Grid system guarantees reliable, fault-tolerant parallel program execution in the Grid based on the parallel checkpoint mechanism developed for P-GRADE.

The GRM/PROVE performance monitoring and visualization toolset has been extended towards the Grid and connected to a general Grid application monitor (Mercury) developed in the EU GridLab project. Using the Mercury/GRM/PROVE system any parallel application launched by the integrated Condor/P-GRADE system can be remotely monitored and analysed at run time. On-line performance and execution visualization support is provided by PROVE.

The new version of P-GRADE under construction supports component based workflow definition and co-ordinated multi-job execution for the Grid. The P-GRADE workflow system supports parallel execution at both inter-job and intra-job level. Automatic checkpoint mechanism for parallel programs supports the migration of parallel jobs inside the workflow providing a fault-tolerant workflow execution mechanism. The same remote monitoring and performance visualization techniques that are used for the single parallel job execution can be applied for the workflow execution in the Grid. The current workflow engine is based on Condor DAGman but in the near future a generic Grid Application Manager will be developed that takes care of all kinds of job and file transfer optimizations during the execution of the workflow.

P-GRADE has been used to develop various parallel applications including an urban traffic simulation program (Gourgoulis et al. 2003) that will be further developed towards a multi-city traffic simulator running in the Grid. Currently, P-GRADE is applied for parallelising large chemical molecule simulations as well as for creating a smog forecast system. The Grid version of the smog alarm system will advice possible strategies for reducing smog pollution.

P-GRADE 8.2.2, the interactive version of P-GRADE can be freely downloaded (with a User's Manual and a set of demo programs) from the www.lpds.sztaki.hu web site. The new release (P-GRADE 8.3) with the single job and workflow Grid execution support is planned by the end of 2003.

6 ACKNOWLEDGEMENT

The work described in the paper was partially supported by the following grants: Hungarian Scientific Research Fund (OTKA) no. T042459, Hungarian IKTA-00075/2001, EU GridLab IST-2001-32133 and EU DataGrid IST-2000-25182 projects.

7 REFERENCES

Balaton, Z. and Gombás, G. 2003. "Resource and Job Monitoring in the Grid"., Proc. Of EuroPar'2003, Klagenfurt, 2003, pp. 404-411.

Balaton, Z., Kacsuk, P. and Podhorszki, N. 2001. "Application Monitoring in the Grid with GRM and PROVE". Proc. of the Int. Conf. on Computational Science – ICCS 2001, San Francisco, pp. 253-262.

Drótos, D., Dózsa, G. and Kacsuk P. 2001. "GRAPNEL to C Translation in the GRADE Environment". In: Parallel Program Development for Cluster Computing, Methodology, Tools and Integrated Environments (eds: Cunha, C., Kacsuk, P. and Winter S.C.), Nova Science Publishers, Inc. pp. 249-263.

Foster, I. and Kesselman, C. 1999. "The Globus Toolkit", in Foster, I. and Kesselman, C, editors, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publisher.

Gourgoulis, A. et al, 2003. "Using Clusters for Traffic Simulation". Proc. of MIPRO'2003, Opatija.

Kacsuk, P. 2000a. "Visual Parallel Programming on SGI Machines". Invited paper, Proc. of the SGI Users' Conference, Krakow, Poland, pp. 37-56.

Kacsuk, P. 2000b. "Systematic Macrostep Debugging of Message Passing Parallel Programs". Journal of Future Generation Computer Systems, Vol. 16, No. 6, pp. 609-624.

Kacsuk, P. 2000c. "Performance Visualization in the GRADE Parallel Programming Environment". Proc. of the Fourth Int. Conf. on High Performance Computing in Asia-Pacific Region (HPC'Asia 2000), Peking. pp. 446-450.

Lovas, R. et al, 2002. "Application of P-GRADE Development Environment in Meteorology". Proc. of DAPSYS'2002, Linz, pp. 30-37.

Thain, D., Tannenbaum, T. and Livny, M. 2003. "Condor and the Grid", in Fran Berman, Anthony J.G. Hey, Geoffrey Fox, editors, Grid Computing: Making The Global Infrastructure a Reality, John Wiley.

MODELING PHYSICS: FROM MATERIALS TO ENVIRONMENTAL PROBLEMS

Lucilla de Arcangelis
Second University of Naples and INFM
Information Engineering Department
Via Roma 29
81031 Aversa (CE), Italy
Email: dearcangelis@na.infn.it

KEYWORDS
Parallel computing, polymers, filtration, fracture.

ABSTRACT

Since several years modeling physics on networks via Monte Carlo simulations has turned out to be a very powerful and efficient method for a number of problems. Lattice models present, in fact, the strong advantage of simple implementation and high computational speed with respect to continuous space models and therefore have been very popular when the interest is on the large scale behavior and the details of the small scale are irrelevant. More precisely, if each element of the lattice represents the system on a mesoscopic scale this approach would allow the simple introduction of heterogeneities, extremely important to study the physical behavior of real systems.
Some examples of different physical problems are presented, all sharing the common feature to have been successfully studied by parallel simulations on networks. The sol-gel transition, for instance, transforms a viscous fluid (the sol) into an elastic disordered solid (the gel). This is done by inducing chemical bonding between monomers in the solution until a macroscopic structure able to bear stresses is created. This transition is therefore characterized by a divergent viscosity and an elastic modulus becoming different from zero in the gel phase. These materials have a complex phenomenology and have been quite intensively studied in the last decades for their relevance in many research fields, from proteins to food industry. The complex macromolecular chain is actually formed by a sequence of chemical units (called Kuhn segments) which are themselves made of several molecules. A simple model has been proposed which considers each monomer as a cell on a lattice, linked to other monomers via bonds of varying length and diffusing on the lattice with random steps obeying local rules. This minimal model has allow the study of both viscoelastic and dynamic properties at the transition, providing numerical results in excellent agreement with the experimental data.
Deep bed filtration is a standard technique employed in industries to eliminate impurities from a flowing fluid. The filter is in general a porous medium where the viscous fluid flows at low velocities, i.e. in a laminar flow. A random network of tubes could easily model a filter and the solution for the local velocities in each tube is efficiently found with standard numerical methods, as conjugate gradient or Gauss-Siedel relaxation. Once the flow field is determined, particles can be injected in the system and propagate under the assumption of perfect mixing at the nodes. More advanced codes take into account the extra pressure in a tube due to the presence of a particle inside, derived via analitical calculation of lubrication theory. This model has allowed the study of the penetration depth ditribution of the particle inside the filter and the relevance of cooperative effects among particles.
Fracture of heterogeneous media is a problem extremely relevant both for structures and in the environmental field. Crack dynamics and morphology are of crucial interest in civil engineering and seismology. Moreover, surface cracking, extremely dangerous for composite materials, has also a strong impact on environmental problems, as drying of grounds. In order to study the mechanical behavior fo a heterogeneous system under external stress, a single element of the lattice models a material grain of mesoscopic size with a given elastic characterisitcs. A random network of such elements is therefore build by assigning the elastic constants or the breaking threshold following phenomenological distributions. Once the external stress is applied, the local stress distribution can be numerically evaluated by solving the Lamé equation. The fracture process can then be modelled by breaking the elements in the system for which the local stress exceeds the breaking threshold. Within this approach, it has been possible, for instance, to determine how the macroscopic fracture stress scales with the size of the system, property of great interest for engineers, of characterize the rupture process for different disorder degrees. Similar models have been applied to study the surface cracking of a thin layer of elastic meterial on a rigid substrate.
Finally, in recent years cellular automata has turned out to give interesting results if applied to study the statistics of seismic events. These models are based on the sand pile model for Self-Organized Criticality and are able to reproduce the scaling behavior of the Gutemberg and Richter law for earthquake magnitude. It would be interesting to introduce similar models to study other statistical properties of seismic catalogues, as the the interoccurrence time distribution, which would allow to predict the occurrence of the next seismic event.
All these problems require intensive computing effort, very performant processors and a large number of resources. The Grid offers the possibility to satisfy these requests, contrary to a single machine approach.

MODELLING METHODOLOGY

SIMULATION LANGUAGES AND MODELS

Extending the capabilities of the SMPL discrete-event simulation language*

X. Molero, V. Santonja, I. Torregrosa and J.A. Alegre
Departament d'Informàtica de Sistemes i Computadors
Universitat Politècnica de València
Camí de Vera, s/n. 46022 València (Spain)
e-mail: xmolero@disca.upv.es

Keywords: discrete simulation, facilities, explicit queues, customer classes, storages

Abstract

SMPL is an open-source simulation language used at universities of many countries both for research activities and in discrete simulation teaching. Despite of that, until now we have not found in the literature significant improvements to its original capabilities. In a previous work we proposed the addition of several instrumentation tools based on those proposed in the CSIM language. In this work we have improved the modeling capabilities of SMPL by adding several features. First of all, we have added facilities with infinite servers to model delays, and most important, customer classes can be defined in the model; the inclusion of classes can be used in order to make explicit the behavior of customers in the model and also to get statistics for each class in reports. Second, we have included explicit queues, storages, and an unqueue function, for both facilities and explicit queues, initially proposed by the creator of SMPL as a natural extension. Finally, the random number generator has been updated to that proposed by L'Ecuyer in order to avoid hardware dependency.

In this paper, we present a brief description of the new features added to the SMPL simulation language, the way they can be used in the model code, and some aspects about their internal implementation.

1 Introduction and Motivation

Simulation is the imitation of the operation of a real-world process or system over time [1]. In particular, simulation is a useful technique for computer systems performance analysis. The choice of the programming language has a significant impact on the timely development of the model. General-purpose languages are portable and provide a good control over the efficiency and run time of the simulation [7]. Moreover, a model developed in a general purpose-language can be easily converted for execution on different computer systems.

SMPL (*Simple Portable Simulation Language*) [9, 10] is a discrete-event simulation language. It is a functional extension of the C general purpose programming language, although implementations in other programming languages also exist, as Fortran, Pascal or PL/I. This extension takes the form of a set of library functions with, together with the programming language itself, compose an event-oriented simulation language. Simulation operations are performed by means of calls on the functions of the simulation subsystem. This approach provides a simulation capability suitable for small to medium scale models, whereas process-oriented simulation languages, like CSIM [4], are preferable for large scale models.

In a SMPL simulation model, there are three kinds of entities: facilities, customers and events [10]. A facility is used to model a resource, such as the disk in a computer system or the network in a distributed system. In static form, a system comprises a collection of interconnected facilities. The interconnection of facilities is not explicit, but rather is determined by the routing of customers between them. Customers represent the active entities in the system. The dynamic behavior of the system is modeled by the movement of customers through the facilities defined in the model. Events represent a change of state of any system entity, such as the arrival of a customer into a facility. A SMPL simulation program comprises an initialization routine, a control routine, and some number of event routines.

Although SMPL was designed many years ago, nowadays it continues being used at the universities of many countries both for research activities and for teaching basic simulation matters due to its simplicity and easy of use. For example, SMPL is included as a part of the queue module of the SimPack toolkit [6]. Moreover, one of the most important features that makes SMPL to be well-known in the scientific community is that source code is available to users. Consequently, SMPL original design can be adapted or modified to meet the needs of each particular system simulation requirement.

Until now, we have not found in the literature significant improvements to that language since its original design proposed by M.H. MacDougall. In a previous

*This work was supported by the Spanish CICYT under grant TIC2000-0472-C03-03

work [11] we have improved the modeling capabilities of SMPL by adding several instrumentation tools. They can be used to collect simulation variables such as queueing times, queueing lengths, or response times. These tools, implemented as a collection of procedures, will help programmers to improve clarity and readability of simulation programs. The way they can be used has been inspired on the instrumentation capabilities of the CSIM simulation language. In particular, we have added four main structures to collect system data: `tables`, `qtables`, `boxes` and `meters`.

In the work presented here we have improved the modeling capabilities of SMPL by adding several new features. First of all, we have added facilities with infinite servers to model customer delays, and most important, customer classes can be included in the model. Classes can be used in order to make explicit different behavior of customers in the model and also to get statistics for each class in reports. Second, we have included explicit queues, storages, and an unqueue function, for both facilities and explicit queues, initially proposed by the creator of SMPL as a natural extension. This unqueue function can be applied to waiting customers waiting both in facilities and in explicit queues. Our implementation also provides results for customer classes for all types of facilities and also for explicit queues. Finally, the random number generator has been updated to that proposed by L'Ecuyer in order to avoid the original hardware dependency from computers based on 80x86 microprocessors.

The rest of the paper is organized as follows. Section 2 reviews the most important aspects of the capabilities added to the SMPL language and section 3 shows some details about their implementation. Finally, Section 4 summarizes the conclusions and future work.

2 Added Features

This section describes a set of extensions for the basic SMPL simulation language. These include some additional queue operations, management of classes for customers, and three new simulation constructs: facilities with infinite servers, explicit queues and storages. In some cases, descriptions presented in [10] were a starting point for our implementation. Names and syntax for the added functions have been inspired jointly on those used in [10, 4]. Table 2.1 summarizes a subset of the functions added to the SMPL language. The entire set of functions and its prototypes are detailed in the `smpl.h` file.

2.1 Facilities

The original set of inspector functions for facilities has been extended in order to get a more complete number of performance indices. In particular, `X(f)` gets the mean throughput of a facility `f`, `R(f)` gets the mean response time, and `Rl(f)` retrieves the number of release operations.

A model may need the ability to remove an arbitrary customer from the waiting queue associated to a facility. To do that we have implemented the `unqueue(f,t,&te)` function. This function makes explicit the release of the facility `f` by the waiting customer `t`[1]. Output parameter `te` represents the remaining event time of that customer when it has been previously preempted. In addition, we have included the `qentry(f,n)` function, which returns the token number of the `n`-th entry in the queue for facility `f`.

Facilities in the original SMPL design only can have a finite number of customers. In order to easily model customer delays we have included a construction representing a facility with infinite servers. The function `facility(s,INF_SERV)` creates and names (string `s`) a facility of this type, and returns a facility descriptor `f` used to specify the facility in other operations. Release and request operations can be made using the same functions which work on facilities with a finite number of servers. However, it should be note that a customer entering a facility with infinite servers is never blocked because he always can find a free server. Also, performance indices can be accessed by means of the same declared functions.

2.2 Customer Classes

When designing complex models it is often necessary to make explicit the behavior of different classes of customers. To do that we have included an attribute to each customer in the model representing its class. In particular, the `def_class(s)` function defines a class in the model and returns an integer identifying it. The `set_customer_class(t,c)` function assigns the class `c` to the customer `t`. The class of a customer can be retrieved using the `customer_class(t)` function.

Customer classes can be used to segregate data for reporting purposes. Class statistics can be gathered for all types of facilities and for explicit queues (see next section). Data gathering needs to be enabled previous to the simulation phase. The `collect_class(1)` function enables data collection in all the facilities defined in the model, whereas `collect_class_station(fq)` can be used to restrict the number of stations (facilities or explicit queues) where statistics are collected considering classes separately.

The original set of functions to retrieve status and performance measures from facilities or explicit queues has been expanded to include the class as an input argument. For example, the `U_class(f,c)` and `R_class(f,c)` functions get the mean utilization and the mean response time of facility `f` for class `c`, respectively.

2.3 Explicit Queues

System models frequently need queues which are not directly tied to facilities; these constructions are explicit queues, or simply queues, where the server elements are not needed. Figure 2 shows a case of queue with 4 customers. A queue is defined by a call to the `queue(s)` function. A customer `t` enters the queue `q` with priority

[1] In this paper words *customer* and *token* refer to the same modeling entity.

Facilities	
`f=facility(s,INV_SERV)`	define a facility with infinite servers & return descriptor
`unqueue(fq,t,&te)`	dequeue token t of facility or queue; te: remaining service time
`t=qentry(fq,n)`	returns token in nth entry of a facility or a queue
`r=X(fq)`	get mean throughput of a facility or a queue
`r=R(fq)`	get mean response time of a facility or a queue
`u=Rl(fq)`	get completions of a facility or a queue
Class Management	
`c=def_class(s)`	define class & return descriptor
`set_customer_class(t,cs)`	set class c to token t
`c=customer_class(t)`	get class of token t
`collect_class(e)`	get statistics for classes at all facilities if e=1
`collect_class_station(fq)`	get statistics for classes at a facility or a queue
`r=X_class(fq,c)`	get mean throughput for class c in a facility or in a queue
`r=R_class(fq,c)`	get mean response time for class c in a facility or in a queue
`r=U_class(fq,c)`	get mean utilization for class c in a facility or in a queue
`u=Rl_class(fq,c)`	get completions for class c in a facility or in a queue
`r=B_class(fq,c)`	get mean busy period for class c in a facility or in a queue
`u=Lq_class(fq,c)`	get queue mean length for class c in a facility or in a queue
Explicit Queues	
`q=queue(s)`	define queue & return descriptor
`enq(q,t,p)`	enqueue token
`t=deq(q)`	dequeue token at the head of the queue
Storages	
`st=storage(s,n)`	define storage of n elements & return descriptor
`a=allocate(n,st,t)`	allocate n elements for token t: a=1 if queued
`deallocate(n,st,t)`	deallocate n elements for token t
`storage_capacity(st)`	number of elements defined
`u=avail(st)`	number of elements currently available
`u=storage_qlength(st)`	number of tokens currently waiting
`u=storage_queue_cnt(st)`	number of queued tokens (cumulative value)
Reporting	
`full_report()`	generate complete report
`report_facilities()`	generate report for facilities
`report_queues()`	generate report for explicit queues
`report_storages()`	generate report for storages

Variables: `t` (token), `f` (facility), `q` (explicit queue), `fq` (facility or queue), `st` (storage), `c` (class), `r` (real), `u` (unsigned integer), `s` (string)

Figure 1: Main set of added functions to the SMPL language.

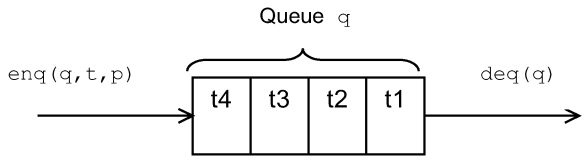


Figure 2: A case of explicit queue

`p` by means of the `enq(q,t,p)` function. Customers in the queue are ordered according to their priority. The `deq(q)` function is used to dequeue the first customer in the queue. Additionally, in order to allow more functionality to the this kind of objects, it is possible to use the `unqueue()` and `qentry()` functions, both explained in Subsection 2.1. Note that, once a customer has entered a queue, it only can release that queue if other customer executes the `unqueue()` or the `deq()` function. A complete report of explicit queues of a model can be obtained by using the `report_queues()` function.

2.4 Storages

A storage is a resource which can be partially allocated to a requesting customer. A storage consists of a counter indicating the amount of available storage and a queue for customers waiting to receive their requested allocation. Figure 3 graphically shows the situation of one storage that limits the number of customers in a region of the model.

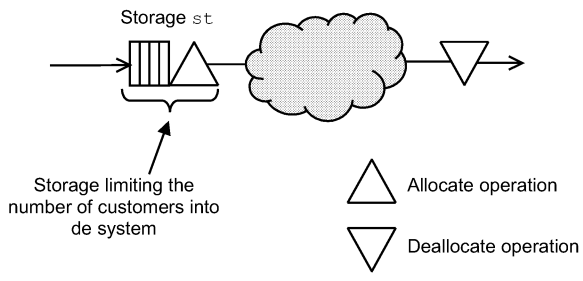


Figure 3: Use of storages in a queueing model

The function `storage(s,n)` builds a descriptor for a storage with a capacity of `n` units. `allocate(k,st,t)` is called to allocate `k` units on storage `st` for customer `t`. If the amount of available storage is sufficient, the amount available is decreased by the requested amount, 0 is returned to the caller and the requesting customer continues; otherwise, the allocation request is queued and 1 returned. `deallocate(k,st,t)` deallocates `k` units on storage `st`; if this frees enough space for a waiting customer, that request is dequeued and rescheduled. Note that waiting customers are ordered in a FIFO policy because the function employed to allocate storage does not use customer priority as an input argument.

A set of inspector functions have been added to storages. For example, `avail(st)` returns the number of units currently available on storage `st`, and `storage_qlength(st)` returns the number of tokens currently waiting for storage. Finally, a complete report of storages of a model can be obtained by using the `report_storages()` function.

2.5 Random Number Generation

We have changed the original random number generator provided by SMPL in file `rand.c` by the MRG32K5A combined multiple recursive generator proposed in [8]. This generator is designed as the combination of two multiple recursive generators of order 5. The resulting generator has an approximate period of 2^{319} and it can generate independent random streams spaced 10^8 random numbers departing from the original set of seeds. The needs of this random number generator is only the use of the double precision IEEE 754 standard for floating point operations, thus being independent on the computer architecture that runs the model.

The use of the `ranf()` function is similar to the original one. In our implementation we have maintained a number of 15 streams departing from stream 1. This number of streams can be easily increased if necessary.

3 Implementation

The implementation of SMPL uses the C language. SMPL basic files are `smpl.c` (main simulation subsystem), `rand.c` (random variate generation functions), `smpl.h` (external name declarations) and `stat.c` (normal and t distribution quantile functions). In order to include the extra capabilities presented in this work, we have added code to the `smpl.c`, `rand.c` and `smpl.h` files.

We have maintained the compactness of its original design, allowing both the use of SMPL in computers with limited memory sizes and as a part of a larger application-oriented simulation system. Compactness has been achieved using a single basic data element for all data structures. These elements comprises five fields, `L1` and `L2` for real values, and `L3`, `L4` and `L5` for integer values. SMPL maintains a pool of these elements, which are allocated from this pool in order to build facility descriptors, queue entries, and event list entries. This pool of elements was previously extended in [11] in order to construct `table`, `qtable`, `meter` and `box` descriptors. Now, additional structures are defined in order to support the new functions. Figure 4 shows these data structures.

In the structure for an facility with infinite servers we store different measures that are needed to obtain performance indices related to this facility (Figure 4a): the time when the last change occurred at the facility, the number of costumers in service, the number of customers served by this facility from the start of the simulation, the accumulated sum of the service times, and the length-time product sum.

Classes of customers are supported through the use of two basic structures. A descriptor for each class (Figure 4b), containing (a pointer to) its name and a class identifier. For each class defined in the model, two additional structures should be added to the descriptor of those facilities and explicit queues where the user needs the performance indices detailed per class. This structure contains information related to a class of customers (Figure 4c): the number of preemptions, number of customers served at the facility, queue length, the accumulated sum of the service times, and the length-time product sum.

The structure for storages has been extracted from [10] with minor changes (Figure 4d).

Finally the structure of explicit queues is shown in Figure 4e. It contains 10 fields, the most interesting are: the accumulated sum of the idle time, the length-time product sum, the number of customers departing from the queue since the beginning of the simulation, and the queue length.

4 Summary and future work

In this paper we have described a set of functionalities that have been added to the well-known SMPL language in order to extend its modeling ability. In particular, we have included unqueue operations in facilities, facilities with infinite servers, explicit queues and storages. Also, class reporting can be obtained for all kind of facilities and also for explicit queues. All these added functions will help model designers to implement simulation programs that are easier to read, debug and understand.

The added source to the SMPL original code is totally open to the community and it can be obtained by sending an e-mail to one of the authors of this work. An extended version of this paper, including some expermimental results when simulating two simple layer-4 scheduling strategies used in web clusters [2, 3], is also available.

Nowadays, we are working on obtaining statistics of classes for storages. Finally, we are also studying the possibility of adapting SMPL in order to consider other scheduling policies at facilities such as round robin.

References

[1] J. Banks (ed.), *Handbook of simulation. Principles, Methodology, Advances, Applications and Practice*,

unused	sum time product	index of name	inf f. chain link	is infinite
last change time	sum time	number in service	number served	unused

(a) Descriptor of a facility with infinite servers.

unused	unused	index of name	class chain link	class id.

(b) Descriptor of a class.

last q. change time	length-time prod. sum	preempt	used class	enable class collect
busy start	busy sum	release cnt	queue exit	queue length

(c) Descriptor of a class in a facility or explicit queue.
Each class appends two lines to the structure of a facility.

last st. change time	space-time prod. sum	deallocation cnt	allocs in storage	queue exit cnt
length-time prod. sum	deallocation amt	index of name	storage chain link	head of queue link
last q. change time	busy time	queue length	space remaining	space defined

(d) Descriptor of a storage.

length-time prod. sum	idle time sum	index of name	queue chain link	head of queue link
last change time	queue exit	queue length	is queue	unused

(e) Descriptor of an explicit queue.

Figure 4: Added elements to the SMPL basic element pool.

McGraw-Hill, 1991.

[2] V. Cardellini, E. Casalicchio, M. Colajanni, and P.S. Yu. The State of the Art in Locally Distributed Web-server Systems. IBM Research Report, RC22209 (W0110-048). October, 2001.

[3] E. Casalicchio and S. Tucci. Static and dynamic scheduling algorithms for scalable web server farm, *Proceedings of 9th Euromicro Workshop on Parallel and Distributed Processing*. IEEE Computer Society. February, 2001.

[4] *User's guide: CSIM18 Simulation Engine (C version)*, Mesquite Software, Inc.

[5] A.M. Law and W.D. Kelton, *Simulation modeling and analysis*, McGraw-Hill, 1991.

[6] P.A. Fishwick, *Simulation model design and execution. Building digital worlds*, Prentice Hall, 1995.

[7] R. Jain, *The art of computer system performance analysis*, John Wiley & Sons, Inc., 1991.

[8] P. L'Ecuyer. Some recommendable uniform random number generators, *Proceedings of 13th European Simulation Multiconference*. The Society for Computer Simulation International, pp. 185–190, 1999.

[9] M.H. MacDougall, *SMPL – a simple portable simulation language*, Amdahl Corp., Technical Report, April 1, 1980.

[10] M.H. MacDougall, *Simulating computer systems. Techniques and tools*, The MIT Press, 1987.

[11] X. Molero, V. Santonja, J.A. Alegre, I. Torregrosa, Adding instrumentation tools to the SMPL discrete-event simulation language, *Proceedings of the EUROSIS Industrial Simulation Conference*, pp. 28–37, 2003.

[12] H. Schwetman and J. Brumfield. Data analysis and automatic run-length control in CSIM18, *Proceedings of the 1997 Winter Simulation Conference*, pp. 687–692, 1997.

ON THE USE OF DOE FOR CHARACTERIZATION OF JAVASPACES

Hans De Neve, Frederic Hancke, Tom Dhaene, Jan Broeckhove and Frans Arickx
Department of Mathematics and Computer Science
University of Antwerp
2020, Antwerp
Belgium
{hans.deneve, frederic.hancke, tom.dhaene, jan.broeckhove, frans.arickx}@ua.ac.be

KEYWORDS
Design of Experiments, Distributed Computing, JavaSpaces, Performance Modelling.

ABSTRACT

As the world of distributed computing evolves quite rapidly, decent statistical techniques to quantitatively test these distributed platforms are needed more than ever. Application dependant platform comparisons are not satisfying enough anymore. This contribution introduces the use of a statistical technique, called Design Of Experiments (DOE), to characterize processes. This technique is being investigated for its application on distributed systems, with JavaSpaces as an example.

INTRODUCTION

In our research group CoMP (*Computational Modelling and Programming*) research is done in the field of computational science, aimed at understanding physics and engineering problems through modern modelling techniques, using new software development paradigms and advanced mathematical techniques.

Many problems, as in the area of quantum physics, often involve very intense and large computations. Therefore, distributed platforms, such as JavaSpaces [FHA99], MPICH [MPI] (an MPI [GES99] implementation), etc. provide a better and faster solution. The goal of our project is to characterise different existing platforms for certain kinds of calculations.

This contribution focusses on a very different approach to characterization than previous attempts [HSD+03]. The use of a statistical technique, called Design Of Experiments (DOE) [Roy01], for characterising distributed systems and predicting its behaviour based on very small experiments, is being investigated. The distributed platform used for the test is JavaSpaces, which is first discussed in short. The part of DOE needed for the rest of this paper will be next, followed by a brief summary of the parameters used to characterise a distributed system, and finally, the first results of DOE applied on JavaSpaces are discussed.

JAVASPACES

JavaSpaces is one of many implementations of the so called Linda Tuple Spaces concept. The idea is that objects can be thrown in a virtual environment, called a space, and taken out, or simply read, by any object connected with that space. Many distributed platforms have been built using this idea. Other well known implementations, besides JavaSpaces, are TSpaces and GigaSpaces.

JavaSpaces was built on top of Jini [Jin, Edw01] as a service of the Jini technology. Its functionality was kept very simple, but nevertheless is very powerful. In fact, there are only three basic actions on the JavaSpace itself:

- **write**: to write an object *into* the space,
- **read**: to read an object from the space, but leaving the object *in* the space, and
- **take**: to take an object *out* of the space.

There is a fourth operation possible, but this one does not really perform on the space: **notify**, which notifies an object of objects being added to the space.

DESIGN OF EXPERIMENTS

Design Of Experiments (DOE) is a statistical technique describing the process to plan an experiment on a certain system. After the execution of the designed experiment it is possible to analyse the results with statistical methods such as Analysis Of Variance (ANOVA) [HT99, Mon01] and regression analysis. These techniques allow us to build a Response Surface Model (RSM) [TWSA97].

The system to be tested is referred to as a process [TSdC+00]. A process is a combination of machines, methods, fysical reactions, human beings and other resources that, given an input and a set of inputfactors, generates an output where the values of a set of outputfactors (also called responses) can be derived. The system acts as a *black box* (see figure 1).

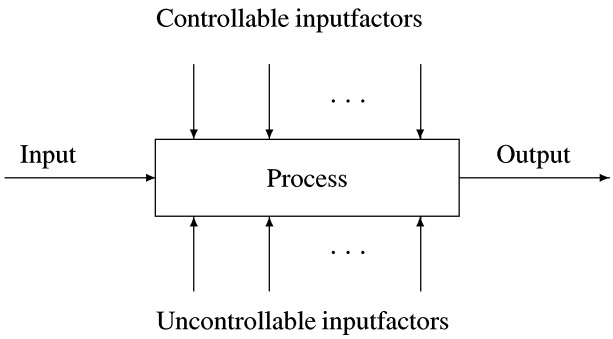


Figure 1: The system to be tested referred to as a process

The inputfactors can be divided in controllable and uncontrollable inputfactors. Controllable inputfactors can be adapted any time, independant of the other factors. For example, the number of machines in a distributed system. Uncontrollable inputfactors can not be adapted independantly any time. For example, the load of the network. Factors can be measurable or not. All controllable inputfactors and outputfactors must be measurable. If an uncontrollable inputfactor is measurable, the experiment can be designed in such a way that this inputfactor (like the load of the network) does not influence the conclusions in the statistical analysis.

Experiment

DOE sets up a complete experiment [HT99, Mon01]. An experiment is a test or a sequence of tests. The levels of the inputfactors will be changed appropriately such that changes of values of outputfactors can be observed and identified. Therefore, these factors must be measurable. After the execution of the designed experiment and the analysis afterwards, conclusions can be drawn on which inputfactors influence which outputfactors and predictions can be made on values of each outputfactor for a set of inputfactors on certain levels.

For each test, each controllable inputfactor must be assigned a level. DOE tries to set up as few tests as possible in an experiment. The choice of the tests is based on replication, randomisation and fixed levels of inputfactors. With replication repetition of every test in an experiment leads to an experimental error. This error is caused by the unknown and uncontrollable inputfactors. Randomisation means the order of execution of tests and its replicates are randomized. With randomisation influences of unknown and uncontrollable factors on the conclusions can be avoided.

Levels of inputfactors for every test can be fixed or random. Using fixed levels, the analysis can only draw conclusions in the area of the fixed levels. This is useful when we are interested in a specific area. Choosing random levels leads to more general conlusions about the area. This is useful when we are interested in the general flow of the output against the input. Because the inputfactors in our experiment are all quantitative, fixed levels will be chosen.

Model

With every experiment a model can be built. A model must be consistent with the target of the experiment. One can not choose a model being too simple, because it may not meet the requirements. On the other hand, one can not choose a model too complex, because too many tests have to be executed. A linear model is a good model to start with. Using a linear model, linear effects and interaction effects of the inputfactors on the outputfactors can be detected and analysed. If the model does not satisfy the requirements, more tests can be executed to build a more complex model. For example a quadratic model.

Full Factorial Design

A full factorial design [HT99, Mon01] is a basic design of experiments with multiple inputfactors. All combinations of levels of inputfactors will be tested. Only two levels for each inputfactor are needed to build a linear model. These two levels are fixed and will be chosen such that an interesting area to model can be covered. Suppose there are m replicates and n controllable inputfactors. Then there will be $m2^n$ tests in the complete experiment. Therefore, this design is called a 2^n full factorial design. Suppose $n = 2$ (with inputfactors called A and B), then the mathematical model for outputfactor Y is:

$$y_{ijk} = \mu + \alpha_i + \beta_j + (\alpha\beta)_{ij} + \epsilon_{ijk} \qquad (1)$$

where $i \in \{-1, +1\}$, $j \in \{-1, +1\}$ and $k \in \{0, 1, ..., m\}$. The low level and the high level are called -1 and $+1$ respectively because there are only two levels. y_{ijk} is the value of the outputfactor Y of the k-th replicate of the test with A on the i-th level and B on the j-th level. μ is the overall mean, α_i is the effect on the i-th level of A on Y, β_j is the effect on the j-th level of B on Y, $(\alpha\beta)_{ij}$ is the interaction effect of A on the i-th level and B on the j-th level on Y, and ϵ_{ijk} is the experimental error of the test.

A regression model can also be built for this experiment. Suppose again $n = 2$, x_A is the value of A and x_B is the value of B. Then

$$y = \gamma_0 + \gamma_1 x_A + \gamma_2 x_B + \gamma_{12} x_A x_B + \epsilon \qquad (2)$$

This model allows us to predict outputfactor Y with inputfactors A and B. The statistical analysis after the execution of the designed experiment will try to fit this model.

To test whether the real system acts as a 2^n full factorial model, one can extend the 2^n full factorial model to a test (with replicates) of the central point. The value of all inputfactors on the central point is on the central level 0. That is, exactly half way between the low and high level. If the response curve in equation 2 contains the output value y_{00k}, then this model is considered satisfying. If not, a more complex model must be built. For example a quadratic model.

Central Composite Circumscribed Design

One possible quadratic model is a 3^k full factorial model. Instead, another quadratic model will be used, called the Central Composite Circumscribed (CCC) model [HT99, Mon01]. The CCC model is especially suited to build Response Surface Models from the regression analysis. Another advantage is that the CCC design includes all the tests of a 2^n full factorial design as well as the central point. The extra tests are those of the starpoints. All inputfactors are set to 0 (the central point) and one inputfactor is set on the new levels $-\delta$ and $+\delta$ ($\delta = (2^n)^{1/4}$). These levels lie outside the levels -1, 0 and $+1$. There are $m(2^n + 2n) + m_c$ number of tests in a CCC design (m_c denotes the number of replicates of the central testpoint). For $n = 2$, the starpoints lie on a circle (see figure 2).

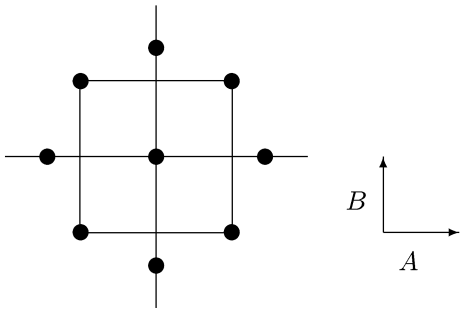


Figure 2: Representation of the testpoints of a CCC model with inputfactors A and B

A regression model can also be built from this CCC design (suppose again $n = 2$):

$$y = \gamma_0 + \gamma_1 x_A + \gamma_2 x_B + \gamma_{11} x_A^2 + \gamma_{22} x_B^2 + \gamma_{12} x_A x_B + \epsilon \quad (3)$$

Quadratic effects can now be detected and analysed, but no quadratic interactioneffects. Therefore, a cubic model must be built.

Analysis Of Variance

Up to this point we have defined two designs and two models. After the designed experiment has been executed, the results can be fitted in the model and the model parameters can be obtained. To fit the model, the Analysis Of Variance with a regression analysis can be used.

Suppose again $n = 2$ and a 2^n full factorial design experiment was executed. Then the ANOVA will test the hypothesis:

$$\begin{aligned} H_0 &: \alpha_i = 0 \\ H_1 &: \alpha_i \neq 0 \end{aligned}$$

with $i \in \{-1, +1\}$ (see equation 1). If H_0 holds, this means inputfactor A has no influence on outputfactor Y. If not (the alternative H_1 holds), this means at some testpoint, A influences Y. Similarly, a hypothesis for β_i and $(\alpha\beta)_{ij}$ will be tested. If the last term equals 0 for all combinations of levels i and j, this means there is no interaction of A and B influencing Y. Another hypothesis supposes $\mu = 0$.

Effect	Sum of Squares	Degrees of Freedom	F-value	P-value
Intercept	SS_M	3	$\frac{MS_T}{MS_E}$	P_M
A	SS_A	1	$\frac{MS_A}{MS_E}$	P_A
B	SS_B	1	$\frac{MS_B}{MS_E}$	P_B
AB	SS_{AB}	1	$\frac{MS_{AB}}{MS_E}$	P_{AB}
Error	SS_E	$df_{tot} - 3$	-	-
Total	SS_T	df_{tot}	-	-

Table 1: Output of ANOVA for a 2^2 full factorial design with replication

To test H_0, a Level of Significance (LoS) must be chosen. This is the risk of concluding H_0 is false, while being true in reality. Usually LoS is five percent. Table 1 shows which hypothesis will be accepted. The first row (intercept or I) indicates $H_0 : \mu = 0$, the second $H_0 : \alpha_i = 0$, the third $H_0 : \beta_j = 0$ and the fourth $H_0 : (\alpha\beta)_{ij} = 0$. If H_0 is true, the F-values (calculated from the Mean Square (MS), Sum Of Squares (SS) and Degrees Of Freedom (DF) and $MS = SS/DF$), should be values from a F-distribution. The P-values indicate the probability the F-value comes indeed from a F-distribuion. If the P-value is less than LoS, H_0 will be rejected and H_1 accepted. The actual test is in fact the comparison of the parameters α_i and ϵ_{ijk}. If the variance of the experimental error is greater than the variance (or influence) of inputfactor A, then H_0 is accepted.

ANOVA can select the significant inputfactors (those where H_0 is rejected). With these factors a Response Surface Model can be built by executing a statistical regression with the least squares method. This regression will result in approximations of the γ_i parameters of equations 2 and 3, such that predictions about the system can be made.

MODEL PARAMETERS

In this section a processmodel is being built to offer the possibility of investigating the use of DOE with JavaSpaces. Therefore, a process with input- and outpufactors must be obtained. The processmodel should be platform and software independant so that it is possible to reuse the model for other spaces, like TSpaces or GigaSpaces, in order to compare these. All factors should also be problem independant.

The processmodel is based on the farmer-worker pattern [KC02]. This pattern is widely used in distributed systems and consists of one farmer and a workers. The farmer divides and distributes a task in b subtasks to workers through a space and collects their subresults. A task is a problem that takes some time to solve on a linear system. Call this time the total theoretical time T_t. Suppose all subtasks have equal execution time c, such that $bc = T_t$. These subtasks will be put in a space. Each available worker will take a subtask from this space, sleep for c time (this is a simulation of the execution of a subtask) and return a dummy subresult to the space. The time between dumping the first subtask in the space and taking the last subresult from the space is called the total time of the distributed system with JavaSpaces (T_{JS}).

Now some inputfactors can be defined. The first inputfactor A denotes the number of workers a, the second (B) the number of subtasks b (such that $bc = T_t$), and the third (C) the execution time c of one subtask. The parameter T_t can not be an inputfactor anymore because it is totally dependant on both b and c. In using sleeps instead of real execution of subtasks, a possible inputfactor processor speed of the worker has been eliminated. An extra supposition is that all subtask can be executed independantly. It is possible to build a more complex processmodel with a variable execution time for every subtask (the subtasks could be beta-distributed, for example) and a communication pattern between subtasks.

T_{JS} may act as an outputfactor. This factor can be normalized because it will ascend when b or c ascend, independant of the behaviour of the system. Therefore, two new outputfactors speedup (S_{JS}) and utilization rate (U_{JS}) are introduced:

$$S_{JS} = \frac{T_t}{T_{JS}}$$
$$U_{JS} = \frac{S_{JS}}{a}$$

U_{JS} denotes the average percentage of a worker being busy executing a subtask. The rest of the time, a worker will wait for a subtask, take a subtask from the space or write a subresult into the space. This is the overhead of the distributed system.

EXPERIMENT

Setup

The experiment has three inputfactors: A (the number of workers), B (the number of subtasks) and C (the execution time of one subtask). The actual values of the levels -1, $+1$ for the 2^3 full factorial design and the levels $-\delta$, -1, 0, $+1$, $+\delta$ are summarized in table 2. A few experiments preceeded this one to determine interessing areas to model with DOE. The starpoints ($-\delta$, $+\delta$) were rounded because all inputfactors could only take discrete values.

Input-factor	$-\delta$	-1	0	$+1$	$+\delta$
A	3	5	8	11	13
B	56	296	648	1000	1240
C	56	296	648	1000	1240

Table 2: Actual values of the inputfactors of the experiment (A: number of workers; B: number of subtasks; C: execution time of one subtask in ms)

The machine of the farmer consisted of an *Intel PIII 733MHz* processor running *SuSE Linux 8.0*. There were thirteen workers. Four workers with an *Intel PIV 1.7GHz* processor, the other nine with an *Intel PIV 1.8 GHz*, all running *SuSE Linux 8.0*. The workers are used at random such that the difference in processor speed will not influence the results largely. On every machine (farmer and workers) *Java SDK 1.4.1* [Jav] and *Jini 1.2.1* [Jin] were installed to run JavaSpaces. The machines were connected with a $100Mbps$ ethernet network. The *Secure Shell protocol version 2* [SSH] was used to start and stop workers automatically.

Results

A 2^3 full factorial design is constructed with $m = 5$ replicates. So, there are $5 * 2^3 = 40$ tests in the experiment. The central point is replicated 10 times, such that $40 + 10 = 50$ tests are executed. For every outputfactor T_{JS}, S_{JS} and U_{JS}, there is a significant curvature. This means that the RSM model does not contain the central point, constructed from the 2^3 full factorial design with 40 tests and that the system does not behave like the linear model. So a CCC design is constructed and the starpoints are tested.

There are $5(2^3 + 2 * 3) + 10 = 80$ tests in the CCC design, 50 tests already being executed in the previous design. Table 3 shows the output of the ANOVA for the outputfactor T_{JS}.

The inputfactors and interactions which are not significant, do not appear in the table. So all P-values have to be lower than LoS (5%). First, notice that the only non-linear term is A^2, so the significant curvature is mainly caused by this term. From this table we can conclude that inputfactors B and C have a linear relation with T_{JS}, inputfactor A has

Effect	Sum of Squares	DF	F value	P value
I	$4.43e+10$	7	46.24	< 0.0001
A	$1.05e+10$	1	76.88	< 0.0001
B	$1.44e+10$	1	105.26	< 0.0001
C	$1.26e+10$	1	92.29	< 0.0001
A^2	$1.62e+09$	1	11.86	0.0108
AB	$1.32e+09$	1	9.64	0.0172
AC	$1.23e+09$	1	9.03	0.0198
BC	$2.56e+09$	1	18.73	0.0034
Residual	$9.58e+08$	7		
Total	$4.52e+10$	14		

Table 3: ANOVA for the total time T_{JS} of the experiment with a CCC design

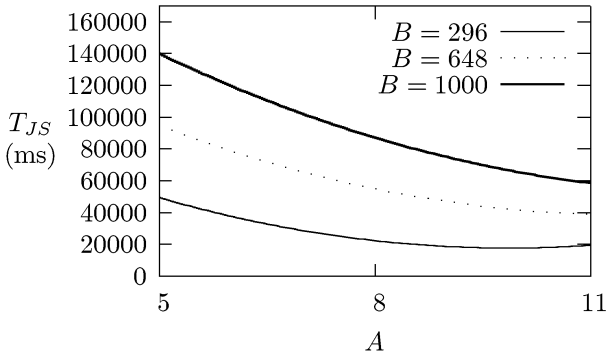


Figure 3: Part of the Response Surface Model of T_{JS} with $C = 648ms$ (A: number of workers; B: number of subtasks; C: execution time of one subtask in ms)

a quadratic relation with T_{JS} and all the linear interactions of the inputfactors influence T_{JS}. Figure 3 shows a part of the RSM that is the result of the regression analysis. The execution time of one subtask (C) is held constant on $648ms$. The X axis denotes the number of workers, the Y axis the total time T_{JS}. Although the points with 3, 5, 8, 11 and 13 workers are tested, a quadratic model between the low level -1 and the high level $+1$ can only be given by interpollation. Globally, as a increases, T_{JS} decreases. This is a logical consequence. The insteresting curve is that of $B = 296$. When there are about 300 subtasks, T_{JS} reaches a minimum around 10 workers. With 11 workers it will take longer to solve the same amount of subtasks. The analysis is able to find this minimum because of the results of the starpoints. At 13 workers the total time will be bigger than at 11 workers.

Table 4, which represents the output of the ANOVA for the speedup S_{JS}, shows that the number of subtasks (B) does not influence S_{JS} at all. The inputfactor A has a linear relation with the speedup, which is again a logical conclusion. The execution time of one subtask (C) has

Effect	Sum of Squares	DF	F value	P value
I	111.95	3	56.47	< 0.0001
A	95.63	1	144.72	< 0.0001
C	10.56	1	15.97	0.0021
C^2	5.77	1	8.73	0.0131
Residual	7.27	11		
Total	119.22	14		

Table 4: ANOVA for the speedup S_{JS} of the experiment with a CCC design

a quadratic relation with the speedup S_{JS}. It reaches a maximum around $850ms$ (see figure 4) for every number of workers. So subtasks which take $850ms$ to execute will give an optimal speedup.

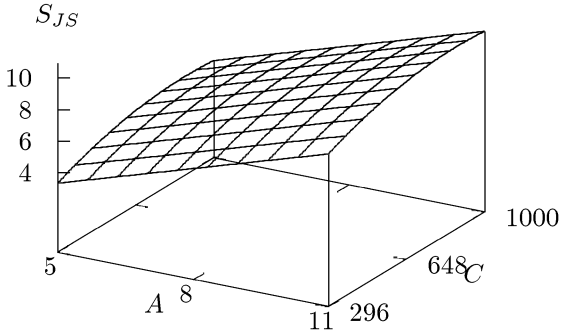


Figure 4: Response Surface Model of S_{JS} (A: number of workers; C: execution time of one subtask in ms)

The utilization rate U_{JS} is the ratio of S_{JS} and A. So, we expect from ANOVA U_{JS} does no longer depend on A but will depend on all the effects of S_{JS}. And this is exactly what ANOVA concludes about U_{JS} (see table 5). Remember that the analysis has no knowledge of the relation between the outputfactors. It analyses every outputfactor independantly. The result of the regression analysis is shown in figure 5. The maximum at $850ms$ for C is again visible.

CONCLUSIONS

We have tried to model JavaSpaces with Design Of Experiments (DOE). An introduction on JavaSpaces and DOE was given, followed by a small processmodel. The processmodel is based on the farmer-worker pattern and contains definitions of inputfactors and outputfactors for a distributed system. These factors have to be problem and factor independant. With that model it is possible to use

Effect	Sum of Squares	DF	F value	P value
I	0.25	2	13.40	0.0009
C	0.16	1	17.02	0.0014
C^2	0.092	1	9.78	0.0087
Residual	0.11	12		
Total	0.37	14		

Table 5: ANOVA for the utilziation rate U_{JS} of the experiment with a CCC design

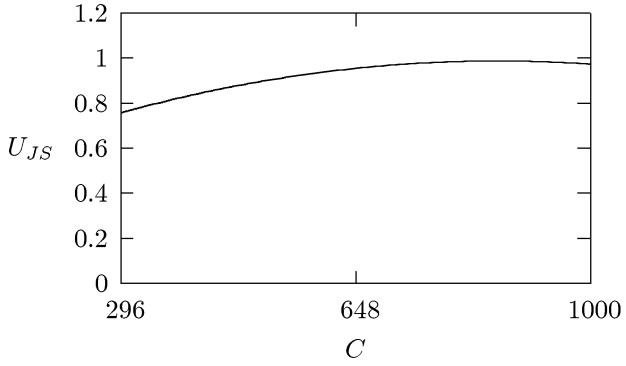


Figure 5: Response Surface Model of U_{JS} (C: execution time of one subtask in ms)

DOE to design an experiment, containing a few tests, on JavaSpaces.

After the execution of the experiment, a regression analysis can be applied. The Analysis Of Variance (ANOVA) selects the inputfactors which influence outputfactors and detects interactions between inputfactors on outputfactors. With these significant inputfactors and interactions a Response Surface Model (RSM) can be built to model the process, the execution of the master-worker pattern with JavaSpaces.

With only a few tests a large area of JavaSpaces can be covered by a quadratic model. Because of the simplicity of the model, it is only a good approximation of the real behaviour of the process in the area between the low and the high levels. If more accuracy is needed, a new experiment has to be designed with a smaller test area (the difference between low and high levels is much smaller). This smaller test area can be located by the knowledge of the results of this constructed model and the targets of the characterisation.

REFERENCES

[Edw01] W. Keith Edwards. *Core Jini*. Prentice Hall, second edition, 2001.

[FHA99] E. Freeman, S. Hupfer, and K. Arnold. *JavaSpaces Principles, Patterns, and Practice*. Addison Wesley, 1999.

[GES99] William Gropp, Lusk Ewing, and Anthony Skjellum. *Using MPI*. The MIT Press, second edition, 1999.

[HSD+03] Frederic Hancke, Gunther Stuer, David Dewolfs, Jan Broeckhove, Frans Arickx, and Tom Dhaene. Modelling overhead in javaspaces. In *Proceedings Euromedia*, pages 77–81. Eurosis, 2003.

[HT99] Charles R. Hicks and Kenneth V. Jr. Turner. *Fundamental Concepts in the Design of Experiments*. Oxford University Press, Inc., 1999.

[Jav] Java. URL: http://java.sun.com.

[Jin] Jini. URL: http://www.jini.org.

[KC02] Rohit Kelapure and Vikrant Colaso. Performance issues in distributed frameworks for large-scale distributed processing. Technical report, Virginia Polytechnic Institute and State University, 2002.

[Mon01] Douglas C. Montgomery. *Design and Analysis of Experiments*. John Wiley and Sons Inc., New York, 2001.

[MPI] Mpich. URL: http://www-unix.mcs.anl.gov/mpi/ mpich/.

[Roy01] Ranjit K. Roy. *Design of Experiments using the Taguchi Approach*. John Wiley & Sons, Inc., 2001.

[SSH] Ssh. URL: http://www.openssh.org.

[TSdC+00] Ledi Trutna, Pat Spagon, Dr. Enrique del Castillo, Terri Moore, Sam Hartley, and Arnon Hurwitz. *Engineering Statistics Handbook*, chapter Process Improvement (Chapter 5). National Institute Of Standards And Technology, 2000. http://www.itl.nist.gov/div898/handbook/pri/pri.htm.

[TWSA97] Patrick N. Koch Timothy W. Simpson, Jesse D. Peplinski and Janet K. Allen. On the use of statistics in design and the implications for deterministic computer experiments. Technical report, Woodruff school of Engineering, Atlanta, Georgia, USA, September 1997.

A SIMULATION MODEL FOR STREAMING APPLICATIONS OVER A POWER-MANAGEABLE WIRELESS LINK

Andrea Acquaviva
Emanuele Lattanzi
Alessandro Bogliolo
ISTI - University of Urbino
Piazza della Repubblica 13
61029 Urbino, Italy
E-mail:acquaviva,lattanzi,bogliolo@sti.uniurb.it

Luca Benini
DEIS - University of Bologna
Viale Risorgimento 2
Bologna, 40136, Italy
E-mail: lbenini@deis.unibo.it

KEYWORDS

Multimedia, Event-Oriented, Resource Management, Real-Time, Simulators.

ABSTRACT

In this work we introduce a hardware-validated simulation model for the exploration of real-time multimedia systems, where system components are modeled as interacting generalized semi-Markov processes (GSMPs). We apply the simulation model to explore the design space of a mobile client accessing streaming data through a wireless network. The model has been characterized and validated against power and performance measurements performed on an instrumented HP's iPAQ with wireless LAN running a MPEG4 video application. We analyze the impact of tuning parameters for the real-time multimedia system (buffer sizes, channel bandwidth, power management policy) on the trade off between power consumption and QoS.

INTRODUCTION

One of the most critical challenges in designing wireless multimedia systems is to provide adequate quality of service (QoS) with optimal energy efficiency. In many cases, there are obvious trade offs between quality of service (e.g., bandwidth, latency) and power consumption. To avoid poorly controlled QoS degradation, multimedia systems are often designed and managed in a conservative fashion, with little consideration for energy efficiency. QoS modeling of wireless networked systems (Cali *et al.*, 1998; Eshghi & Ekhakeem, 1998) is a mature and highly active discipline (refer to (MSWIM, 2002) for an up-to-date overview of the topic), and various approaches have been explored to enhance QoS-oriented models with power consumption models (Raghunathan *et al.*, 2002; Sinha & Chandrakasan, 2001; Marculescu *et al.*, 2001; Krashinski & Balakrishnan, 2002; Zorzi & Chockalingam, 1998). Most of the approaches explored in the past adopted a stochastic discrete event model (Lee & Sangiovanni-Vincentelli, 1998), where the system evolves in an enumerable set of time instants and transitions are randomized.

Our work moves from this widely adopted modeling framework and pushes it one step closer to practice. We present two original contributions. First, we developed, consistently with previous work, a detailed stochastic discrete-event model of a complex multimedia system. Our model includes all hardware and software components involved in communication of multimedia data over a wireless channel: application software, transport and network stack, operating system drivers, power management software, network interface card hardware, wireless channel, base station hardware and software. Many of the model components are parameterized, in an effort to represent a large design space of different hardware and software configurations. Second, we have fully characterized the power and performance metrics in the model with experimental measurements on fully operational real-life hardware and software. As a result, our model is detailed enough to represent a number of complex effects that impact power and quality of service in real-life multimedia systems, while at the same time our characterization flow is precise enough to obtain power and performance estimates that closely match the measured data. Our results demonstrate that the performance and energy consumption of a multimedia system are deeply influenced by a number of interacting hardware and software components, and that focusing only on a part of the system (e.g., the wireless channel or the operating system), could lead to serious design pitfalls.

MODELING POWER-MANAGEABLE REAL-TIME MULTIMEDIA SYSTEMS

At the high level of abstraction, electronic systems exhibit different operating modes and make transitions among them at discrete points in time. Hence, they can be suitably represented as *Discrete event systems* (DES) (Cassandras, 1993). We model real-time power-manageable systems as DES composed of interacting state machines. Each component has a state structure that represents its operating modes and the transitions among them. State transitions are triggered by events

that can be either generated within the component according to some distribution (internal events) or received in input (external events). The number of states of the model of each component may be finite, discrete, or continuous. Infinity states are modeled by means of a finite number of parameterized states. The evolution of the system is described by specifying next-event and next-state functions for each component. Next-event function determines the next triggering event generated by the component, while next-state function determines the destination state of next transition. We model each component as a generalized semi Markov process (GSMP) with non-deterministic next-event and next-state functions based on conditional residual-time distributions and on conditional next-state probabilities (Glynn, 1989). Any GSMP component is composed of a state_structure and a clock_structure. The state_structure component is a Stateflow model that takes in input both internal events (generated by the local clock structure) and external events (coming from an input port) and generates a timeout value corresponding to the residual time of the next triggering event. The clock_structure takes in input a timeout and a reset signal and generates an event when the timeout has elapsed. The interface of the component is specified by means of input/output ports, used to exchange both events and parameter values. The interaction among multiple GSMP components is simply obtained by connecting their input/output ports, as shown in Figure 1. Additional output signals can be used to observe the system behavior, to evaluate cost/performance metrics and to exchange data among modules.

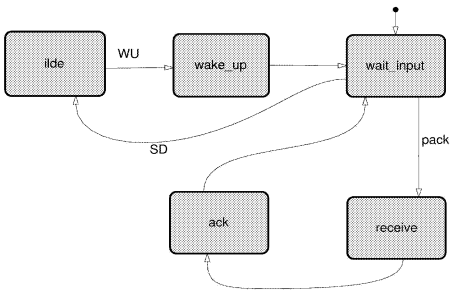


Figure 2. State diagram of the power-manageable NIC.

STREAMING VIDEO OVER AN IEEE 802.11b LINK

The video server runs on a notebook computer directly connected to an access point (AP) through a dedicated wired link. The AP acts as a bridge to the wireless network interface card (NIC) installed on a HP's IPAQ palmtop. The palmtop runs the client MPEG4 decoder that reads, buffers, elaborates and plays-back at a constant rate (consumer rate) the video information read from the network. In order to match real-time video constraints, the client application buffer should be never empty when the decoder looks for a frame to decode. If this happens, a deadline miss occurs. Without any power control, the shape of the traffic on the wireless channel reflects the server transmission rate. If the 802.11b MAC-level protocol power management (LAN/MAN Standards Committee of the IEEE Computer Society, 1999) is enabled, the AP performs traffic reshaping by bufferizing incoming packets to allow the NIC to sleep for a determined period of time. In this period, energy can be saved by the wireless interface. After the expiration of the sleeping period, the card wakes up and asks to the AP if there are packets accumulated to be received. If this is the case, the AP sends the back-log to the card. The traffic reshaping function performed by AP imposes bufferization both at producer and consumer sides.

Model

The block diagram of the Simulink model of the system is shown in Figure 1.

Producer. The producer generates output events (representing network packets) according either to a given distribution of inter-arrival times, or to a trace of time-stamped packet information. For each packet, three properties are either randomly generated or read from the trace: the size (in bytes), the frame it belongs to, and the total number of packets representing the same frame. Packets belonging to the same frame are generated as a burst, while packets belonging to different frames are generated at a rate depending on the application. Packet information is made available at the output ports, together with the event that represents the generation of a new packet.

Base station buffer. Packet events generated by the producer become input events for the buffer of the base station, which is explicitly represented in the model as a limited FIFO queue with a customizable size. The content of the queue is saved in memory as an array of packets each with the corresponding information (size, frame number, packet per frame). If the producer tries to send a packet and the queue is full, a lost event is generated and the corresponding packet is discarded.

Base station. The base station (AP) gets the packets from the queue and send them to the wireless channel. If the input buffer is empty or the receiver in not ready, the AP goes to a waiting state. If the receiver is sleeping because of DPM, the AP goes to an idle state, causing incoming packets to accumulate in the input buffer.

Wireless Channel. The wireless channel is represented by a block that receives input events representing incoming packets and generates, with a given latency, output events representing packet delivery. The wireless channel is bidirectional and it has a user-defined packet-loss probability. Lost packets are not delivered to the receiver. Notice that we use a simple channel model since we are interested in modeling the entire wireless system, rather than the wireless channel by itself. Channel latency, loss probability and bandwidth (implicitly modeled by the receiver) are sufficient to perform realistic system-level simulations. Nevertheless, any channel model (Chong, 2003) can be easily embedded in this block to take into account complex error statistics.

Wireless Network Card. This is the most critical block of the system, since its power consumption is critical for the battery lifetime of the palm-top. The state diagram is reported in Figure 2. It consists of 5 states: idle, waking up, waiting,

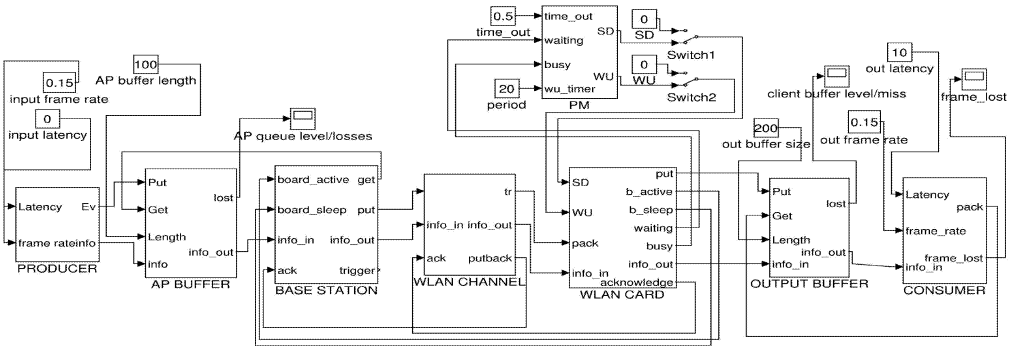


Figure 1. Simulink model of the streaming application over a wireless channel.

receiving, acknowledge. The card is normally waiting. When an incoming packet is detected, the card goes into the receiving state and generates a busy output event to indicate that it cannot receive more packets until the actual one has been processed. The card remains in the receiving state for a time interval that is dynamically computed as the ratio between the size of the packet (including the payload and the protocol overhead) and the bandwidth of the channel. After each packet has been completely received, the NIC takes some time to process the packet at the MAC level and then sends an acknowledge back to the AP through the wireless channel. After the acknowledge packet has been sent, the NIC goes back to the waiting state until the next packet arrives. The timing diagram of the wireless channel is shown in Figure 4. When the power management is enabled, the card can go from the waiting state to a low-power idle state. In our model, this transition is triggered by an input event generated by an external power manager, described later in this section. Wake-up from idle is triggered by the power manager when a determined sleep time has elapsed. Wake up transitions may take a non-negligible amount of time and power, modeled by a separate state.

Power Manager. The power manager does not represent a hardware unit, but the model of the actual implementation of the power management protocol of the 802.11b standard (LAN/MAN Standards Committee of the IEEE Computer Society, 1999). Based on external settings (that can be decided by the user), the power manager generates output events (ShutDown and WakeUp) to notify the beginning and ending of sleeping periods. After the card has been woken-up, it starts to receive packets accumulated by the AP. After each received packet, the power manager resets a timeout counter. If the timeout expires before the reception of a new packet, the card is put again in the idle state by means of a ShutDown event.

Output Buffer. This block represents the bufferization performed by the consumer. It can be used to model either the protocol stack buffer or the application buffer (if present). More levels of buffering can be added. In our case we decided to represent only the UDP protocol buffer, since the application buffer is usually larger and less critical.

Consumer. The consumer simulates a streaming application that reads packets from the output buffer at a given rate. Since real-time constraints impose a constant frame rate, but each frame may be encoded using a different number of packets, packet requests from the consumer do not arrive at a constant rate. Rather, the consumer decides how many packets to read within a frame period based on the information associated with the incoming packets. Frames that either arrive late with respect to the deadline or are incomplete because of a packet loss are discarded by the consumer.

EXPERIMENTAL RESULTS

To perform our characterization experiments, we used a HP's IPAQ palmtop computer. With the IPAQ, we used two different wireless network interface cards for our experiments, CISCO Aironet 350 Series (Cisco System, Cisco Aironet 350 Series Wireless LAN Adapters, 2003) and COMPAQ WL110 (HP WL110, 2003), hereafter denoted by CISCO and COMPAQ, respectively. The power consumption of the NICs was

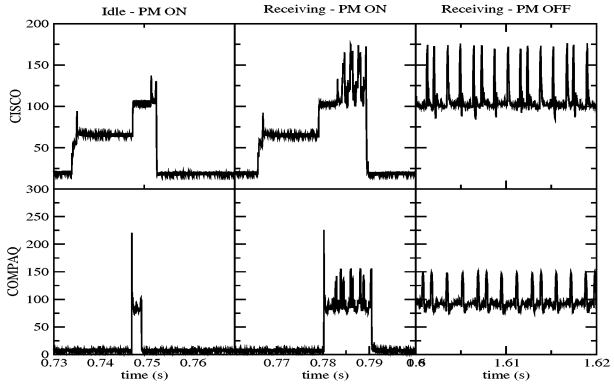


Figure 3. Current profiles obtained under different workload and DPM conditions.

measured using a *Sycard Card Extender* that allowed us to monitor the time behavior of the supply current drawn by the cards.

Characterization

We performed three sets of experiments to characterize: i) the power states of each NIC, ii) the effective bandwidth of the wireless channel and iii) the buffer size of the base station.

Power States. The power states of the NIC were characterized in terms of power consumption and transition time by looking at the power profiles under different workload conditions and DPM settings. Typical results are shown in Fig-

	Wait	Rx	Tx	Sleep	WakeUp
CISCO	495	650	870	100	325
COMPAQ	375	450	750	35	-

Table 1. Power consumption of the NICs in mW

ure 3. The power consumption of CISCO is higher than that of COMPAQ in all states (sleep, idle, receiving, transmitting) but during wake up. Moreover, differently from COMPAQ, CISCO has a non-negligible wake-up time. As for performance, COMPAQ has a slower reaction to incoming packets, as shown in the right-most graphs of Figure 3 obtained while receiving the same burst of packets. Both cards stay idle for a very short time among packets, reception takes most of the time, while higher peaks correspond to the acknowledge transmitted upon reception and processing of each packet. The acknowledge peaks produced by COMPAQ look delayed and wider with respect to those produced by CISCO, actually reducing the number of packets received in a time unit (e.g., in the 20ms time window shown in Figure 3, COMPAQ receives 14 packets while CISCO 15). The measured average power consumption is reported in Table for all power states of the two NICs. The wake-up time of CISCO is 12ms, while that of COMPAQ is neglected.

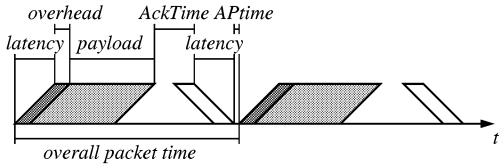


Figure 4. Timing diagram of the wireless channel

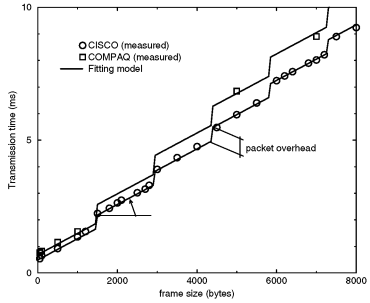


Figure 5. Transmission time as a function of the frame size.

Channel bandwidth. The effective bandwidth provided by the wireless link depend on traffic fragmentation in packets. In fact, each packet has a two-fold overhead: the headers and tails introduced by the protocol stack, and the acknowledge time. To characterize the effective bandwidth by taking packet overheads into account we measured the total time required to send 50 UDP frames of a given size from the laptop to the palmtop PC. The number of frames transferred was chosen small enough to avoid the saturation of the AP buffer. Figure 5 shows the transmission time per frame as a function of the frame size. Steps occurring every 1500 bytes are due to fragmentation. In fact, 1500 bytes is the maximum payload of the Ethernet packets transferred from the laptop to the base station. The actual transmission time can be expressed as:

$$t(size) = \frac{size}{Bw} + (int(\frac{size}{1500}) + 1)(\frac{100}{Bw} + 2 * L + AckTime) \quad (1)$$

where 100 is the number of protocol bytes per packet, L is the channel latency, AckTime is the overall reaction time of the NIC. Notice that both the AckTime and the channel Latency are due to the acknowledge needed after each packet, as shown in Figure 4. Using Equation 1 as a fitting model for the experimental results we can obtain indirect measures of the fitting parameters Bw and L. Fitting curves are plotted in Figure 5. Finally, the channel latency depends on the distance between the base station and the NIC. For our experiments we used a fixed distance of 50cm, providing an ideal channel quality, in order to evaluate the frame loss due to DPM only.

Buffer size. While the size of the UDP and application buffers on the palmtop can be set by the user, the size of buffer used by the base station is constant. We characterized the behavior of the internal buffer by sending bursts of packets across the wireless channel and measuring the packet loss as a function of the packet size and number. For a given packet size, we call *limiting burst size* the maximum number of packets in a burst delivered without any loss. Interestingly, the limiting burst size grows with the packet size: a few packets are lost if more than 100 packets of size 10bytes are sent across the wireless channel, while up to 500 packets of size 1000bytes can be sent without packet loss. The reason for this counter-intuitive behavior is two-fold. First, the internal buffer of the base station is organized in packets, so that it can contain a given number of packets regardless of their size. Second, according to the results outlined in the previous section, the larger the packets the larger the effective bandwidth provided by the wireless channel. Since the internal buffer of the base station saturates because input data (provided by the laptop) are faster than output data (sent across the wireless link), the higher the output bit rate the longer the time required to saturate the buffer. Experimental results indicate that the base station can bufferize up to 100 packets.

Model validation

We validated our model by comparing simulation results and measurements obtained by running two MPEG4 benchmarks: *conference* and *fireworks*. The first one has a frame rate of 15 frames/sec and is composed of 899 frames. The second has a frame rate of 30 frames/sec and is composed of 522 frames. Each benchmark was ran and simulated using both NICs, with all available DPM configurations, namely, COMPAQ PM OFF, COMPAQ PM ON 200ms, COMPAQ PM ON 100ms, CISCO PM OFF, CISCO PM ON 200ms. In order to make simulation results directly comparable with measurements we implemented a trace-based producer generating packets according to a time-stamped trace collected while running the streaming application on the laptop. Experimental results reported in Table show that the average power consumption provided by the simulation model was always within 4% from measurements.

	CONFERENCE		FIREWORKS	
	meas	sim	meas	sim
COMPAQ PM OFF	380.31	377,1	383.74	382.7
PM ON 200ms	56.48	55.5	143.21	139.5
PM ON 100ms	61.66	63.2	146.77	141.9
CISCO PM OFF	497.11	501	502.45	503.6
PM ON 200ms	129.85	133.5	197.3	201.5

Table 2. Power consumption of the NICs in mW

	period	latency	avg pw	AP buf	fr. lost
units	(ms)	(ms)	(mW)	(pac)	(perc.)
PM OFF		100	377.1	1	0%
PM ON	200	100	55.5	4	18%
	200	200	55.5	4	0%
	300	100	52.17	5	47%
	300	200	52.17	5	11%
	300	300	52.17	7	0%
	400	200	52.17	7	52%
	400	300	48.51	7	8%
	400	400	48.51	7	0%
	500	200	47.28	9	61%
	500	300	47.28	9	43%
	500	400	47.28	9	7%
	500	500	47.28	9	0%
	1000	800	42.83	17	29 %
	1000	1000	42.83	17	0%

Table 3. Exploration for different PM configuration and latency values

Design space exploration

The simulation model was used to evaluate the impact of DPM settings not yet supported by the real NICs. In particular we performed experiments on the conference benchmark for different durations of the sleeping periods and for different values of the consumer latency. Results are reported in Table .

For a determined sleeping period, we repeated the experiments by changing the initial latency of the consumer. By increasing the sleeping period, the average power consumption decreases at a cost of higher frame loss. However, the frame loss probability can be reduced by increasing the initial latency, allowing the consumer to bufferize more packets before starting playback. Since the frame consumption rate is constant, the initial bufferization allows the consumer to compensate for the long inter arrival times imposed by the power management policy. In fact, as shown in the table, for a determined sleep duration, the frame loss decreases as a function of the latency. In particular, the packet loss is null whenever the initial latency is greater or equal than the sleep period, since in this case the application buffer is never fully depleted because of DPM. In practice, client-side buffering provides the opportunity for saving power without impairing quality of service, at a cost of an initial latency. The limiting sleep time (and latency) depends on the size of the application buffer. For our case study, power consumption could be reduced to 42.83mW without violating real-time constraints, with a sleep period and a latency of 1sec and a maximum occupancy of the application buffer of 17 packets. This is a 22% improvement over the power savings provided by the longest sleep period supported by the real NIC.

References

Cali, F., Conti, M. & Gregori, E. (1998) Ieee 802.11 wireless lan: capacity analysis and protocol enhancement. *INFOCOM,* , 142–149.

Cassandras, C. G. (1993) Discrete event systems: modeling and performance analysis. *Aksen,* .

Chong, C. C. (2003) A new statistical wideband spatio-temporal channel model for 5-ghz band wlan systems. *IEEE Transactions on Selected Areas in Communications,* **21** (2), 139–150.

Cisco System, Cisco Aironet 350 Series Wireless LAN Adapters (2003). http://www.cisco.com/univercd/cc/td/doc/product/wireless/airo_350/350cards/index.htm.

Eshghi, F. & Ekhakeem, A. (1998) Performance analysis of ad hoc wireless lans for real-time traffic. *IEEE Journal on Selected areas in Communications,* **21** (2).

Glynn, P. W. (1989) A gsmp formalism for discrete event systems. *IEEE Proceedings,* **77** (1), 14–23.

HP WL110 (2003). http://h18004.www1.hp.com/products/wireless/wlan/wl110.html.

Krashinski, R. & Balakrishnan, H. (2002) Minimizing energy for wireless web access with bounded slowdown. *Procedings of MOBICOM,* .

LAN/MAN Standards Committee of the IEEE Computer Society (1999). Part 11: Wireless LAN MAC and PHY Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band.

Lee, E. A. & Sangiovanni-Vincentelli, A. (1998) A framework for comparing models of computation. *IEEE Transactions on CAD ICAS,* **17** (12), 1217–1229.

Marculescu, R., Nandi, A., Lavagno, L. & Sangiovanni-Vincentelli, A. (2001) System-level power/performance analysis of portable multimedia systems communicating over wireless channel. *Procedings of ICCAD,* , 207–214.

MSWIM (2002) *ACM International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems.*

Raghunathan, V., Schurgers, C., Park, S. & Srivastava, M. (2002) Energy-aware wireless microsensor networks. *IEEE Signal Processing Magazine,* **19** (2), 40–50.

Sinha, A. & Chandrakasan, A. (2001) Dynamic power management in wireless sensor networks. *IEEE Design and Test of Computers,* **18** (2), 62–74.

Zorzi, M. & Chockalingam, A. (1998) Energy efficiency of media access protocols for mobile data networks,. *IEEE Transactions on Communications,* **46** (11), 1418–1421.

DISTRIBUTED SIMULATION IN PRODUCTION

DISTRIBUTED MODULAR SIMULATION OF MECHATRONIC SYSTEMS

Alfonso Gambuzza
Oliver Oberschelp
Mechatronics Laboratory Paderborn - University of Paderborn
Pohlweg 98, D - 33098 Paderborn, Germany
E-mail: {Oliver.Oberschelp, Alfonso.Gambuzza}@mlap.de

ABSTRACT

The design of **complex mechatronic systems** has become so intricate that it can only be done by means of computer-aided modelling (Heimann et al. 2001). The models comprise **modules and hierarchies** that are derived from the **physical-topological structure** of the system. For subsequent symbolic and numerical processing the models are transformed into a shape fit for processing, their modular-hierarchical structure having to be observed (Zanella and Stolpe 1998).

A **holistic, consistent,** and **structured** concept for the design of mechatronic systems requires an exact preservation of the system structure from the model to the hard- and software on the processing level. When the design methodology is consistent, it allows a flexible adaption of the model to various scenarios, e.g., to a parallelized off-line simulation or distributed **h**ardware-**i**n-the-**l**oop **s**imulation (HILS), i.e., the direct allocation of information-processing components to physical aggregates used in mechanical engineering.

In the following, we will present in detail the above concept for the modular decomposition of models on the basis of the modular-hierarchical block structure. We will also show the module concept for the structuring of mechatronic systems to be maintained in the information processing. To exemplify this application, we employ the complex model of a coach with an active suspension.

INTRODUCTION

A holistic design of mechatronic systems centers upon an interdisciplinary work based on computer-aided methods and procedures. In the process, distributed, parallelized information processing, used both in computer-aided analysis and synthesis procedures and on HILS testbeds as well as in the digital realization of control systems, gains increasing significance (Schumacher 2002). The focus of our work lies on the structured design of mechatronic systems, from modelling to hardware-in-the-loop simulation to the finished product. In the process, the aim is to design the modular-hierarchical structure of the technical system and the information processing and to keep it consistent throughout every design step. Tools for the mechatronic design have to meet this demand.

There are already several tools supporting such a consistent design process: The dSPACE company (Kiffmeier 1995) realizes distributed real-time implementations of applications in control engineering by means of automatic code generation on the basis of block diagrams (RTI-MP) (dSPACE 1995). Decomposition of the model, however, is restricted to the highest hierarchical level; a hierarchical decomposition is not possible. With its design environment ASCET-SD (ETAS 1996) the ETAS company presents a similar concept. Systems can be modelled on the block-diagram and state-chart levels. Just as with dSPACE's RTI-MP, the user allocates the computing modules to the processors "by hand".

The modular-hierarchical structure that is characteristic of complex technical systems can be employed to structure the information processing. The basic concept here is to couple technical aggregates and local information processing. These mechatronic function modules can be interlinked in analogy to the technical structure. This procedure offers many advantages:

- The structure of the mechatronic system is maintained also on the information-processing level.
- Subsystems can be tested or simulated individually because e.g. the plant model of the technical system can easily be replaced by a technical coupling (**h**ardware-**i**n-the-**l**oop).
- Individual modules or aggregates can be reused as actual aggregates themselves or as a model, including the information processing that controls the system.
- The design process, up to the final product, can be consistent.
- The inherent parallelism in mechatronic systems is taken into account (Honekamp 1998).

If a technical system is already existing in the shape of a modular-hierarchical block structure, it will be possible to use these block structures also for decomposing the model. The decomposition presented in the following is performed by means of annotating the block structure in the model.

STRUCTURING OF MECHATRONIC SYSTEMS

A decomposition of submodels according to the structure of technical systems presupposes an appropriate build-up of the mechatronic system and of the corresponding information processing. The model of the technical system to be set up in the modelling process comprises several features that will be examined in the following: these are above all the modular-hierarchical structure and the function of the technical systems.

These features resp. functions are described differently in the different domains. For instance, mechanical components are developed in a multi-body representation, electrical connections by means of block diagrams, controls in the shape of a mathematical block in state-space representation, etc. The common basis of these domains is mathematics (Gausemeier and Lückel 2000). This requires an algorithmic representation that can either be simulated directly or allow a derivation into a programming language.

MODULAR-HIERARCHICAL STRUCTURE

Modularization and hierarchical structuring are important aids for the solution of the complexity problem in technical systems. Modularizing the function results in function groups. In mechanical engineering this approach is realized as an aggregation. It allows aggregates to be designed, developed, tested, and manufactured independently of the entire system. Thus the overall functional problem is reduced to isolated sub-problems. Only in this way can products be developed and manufactured by different working teams and different suppliers.

An object hierarchy based on function orientation leads to functional groups from which assembly groups can be derived. Assembly groups consist mainly of actuators and sensors, but may also include subassemblies of actuators or actuator/sensor groups. Such groups make up Mechatronic Function Modules (MFM). They consist of actuators, sensors, and information processing along with the hardware required to implement the control system. An MFM itself can include other sub-MFMs and thus form its own hierarchy in which one MFM is an actuator to another.

The entire system differs from its assembly groups in that it operates autonomously. This is made possible by the supporting structure which combines all submodules and thus makes possible new functions which cannot be assigned to a specific aggregate or an MFM. In order to function autonomously, an Autonomous Mechatronic System (AMS) must be in a position to interact with the environment, which means that it must be able to react to changes in the environment. If mechatronic systems have to function autonomously and interact with other mechatronic systems, functionalities such as communication and coordination (including common planning) are required. Control functions, such as the distance control in a convoy, can also be implemented on this functional level. As these functions cannot be allocated to a single autonomous system, it is necessary to have a level in the object hierarchy where different autonomous systems can be cross-linked. A cross-linked group of different AMSs - e.g., a car convoy - makes up a so-called Cross-linked Mechatronic System (CMS). There is no longer a physical link between the autonomous systems, the cross-linking being obtained just by an exchange of information.

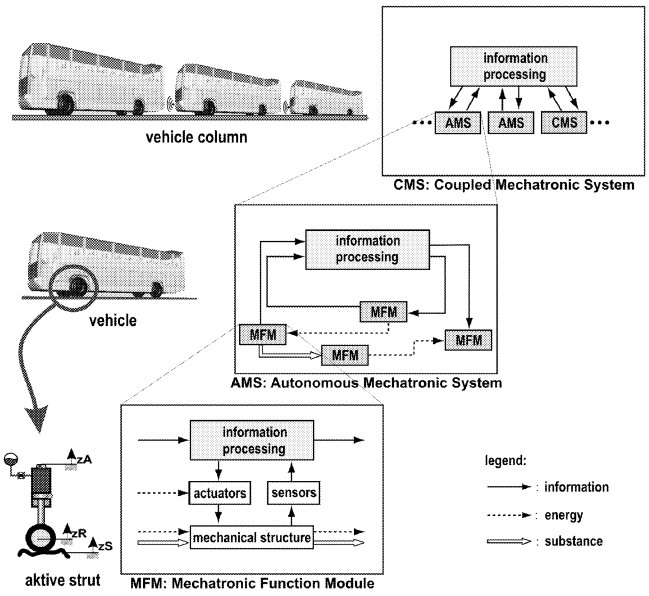


Figure 1. Modular-hierarchical structure

The object levels of the mechatronic aggregations MFM, AMS, and CMS make up an object model for the structuring of mechatronic systems, this structuring being consistently applicable to every stage, from the early function-oriented design to the final technical product. Decomposition is effected by means of a three-level object hierarchy which again classifies three kinds of function modules. On each of these object levels there can exist several function modules which can be interlinked horizontally or vertically.

DECOMPOSITION OF SYSTEM MODELS

On the basis of the modular-hierarchical structuring presented above we will now present some examples of the decomposition of system models.

Every function module can be transformed into a submodel (Fig. 2); the resulting structure is similar to the **tree structure** in the modular-hierarchical description of the mechatronic system:

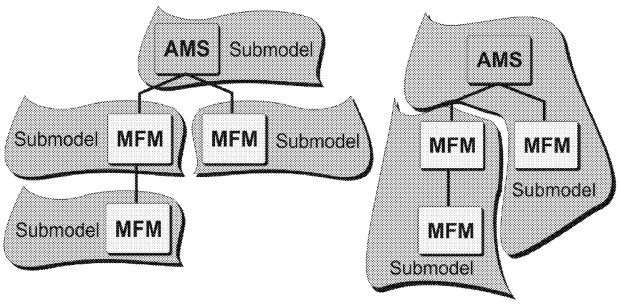


Figure 2. Examples of the decomposition of a model according to the module structure

There is a close relation between the hierarchical positions of a submodel and its **time constraints.** The lower the hierarchical level that the subsystem is on, the smaller the time constants (Lückel et al. 2001). Fig. 3 displays an example of the distribution according to the time constants:

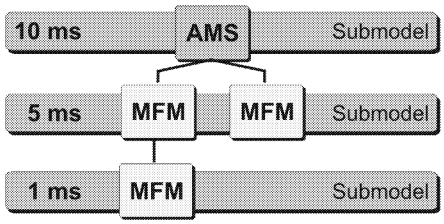


Figure 3. Example of a decomposition according to the smallest time constants of the system model

In combination with these decomposing strategies, hardware- or software-in-the-loop simulations require an additional decomposition of the function modules (Fig. 4). In the process, a function module is decomposed into a plant model and information-processing components, thus making it possible to allocate the modules directly to the hardware components:

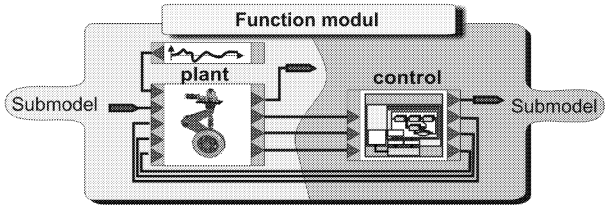


Figure 4. Example of a decomposition according to plant model and information-processing components

DECOMPOSITION OF THE MODEL INTO INDEPENDENT SUBMODELS

According to our proposal, the decomposition of an entire model brings about new submodels. For this purpose one transforms the original model by removing the components of the submodels and exchanging them for references (Fig. 5). This procedure was realized with the CAMeL-View modelling tool (Hahn and Koch 2000) as follows:

- decomposition of the model into subcomponents,
- computation of the evaluation order of the distributed model,
- derivation of the communication configuration from the couplings of the submodels,
- generation of model code for every submodel.

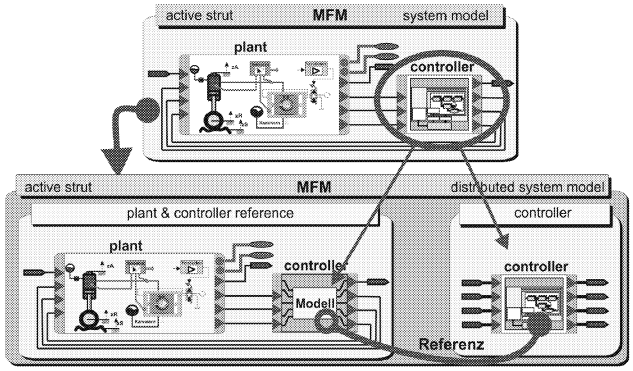


Figure 5. Transformation of the system model into a distributed system model

Fig. 5 presents an example of model decomposition. The overall system model "active strut", which consists of the sub-models "plant" and "controller", is separated into two modules according to the technical structure. Now it is possible to simulate the entire system e.g. on a parallel machine or to test the controller independently in a hardware-in-the-loop environment with the technical system. It is also possible to use the plant module in a software-in-the-loop simulation with an automatic controller to be tested. So the same model-code may be used with different applications (Gambuzza 2002).

REPRESENTATION OF THE DISTRIBUTED MODEL ON THE RUNTIME PLATFORM

For the simulation of a distributed model on different computer platforms it is helpful to make the model implementation itself independent of the respective operating system. This has the advantage that the code generation of the distributed model need not be adapted to the respective environment. In our example, this is due to the open runtime environment IPANEMA (**I**ntegration **Pla**tform for **Ne**tworked **M**echatronic **A**pplications) (Honekamp 1998).

IPANEMA is a platform concept for distributed real-time simulation. It allows a modular-hierarchical organisation of tasks or processes on distributed hardware. IPANEMA is encoded in ANSI-C, thus increasing independence from the platform. The current implementations are based on the

Motorola PowerPC architectures MPC750 and MPC555 as well as Intel x86 (only off-line). As operation systems Windows, Linux, DREAMS (Ditze 1998), and RT kernel (dSPACE) are supported. IPANEMA encapsulates every task of a distributed simulation by means of objects. Objects of the **Calculator** class implement the actual simulation kernels for the corresponding submodels. They do not comprise any administrative or data-management functionality. These tasks fall to the so-called **Assistant** objects that are assigned to one calculator each in order to reduce the calculator's load. This arrangement realises a strict separation between hard and soft real time.

Assistants encapsulate the calculators against access by the so-called **Moderator** object of which just one instance exists and serves as an interface between the application and the user. The moderator coordinates the activities of the calculator/assistant couples. Typical actions are the simulation start, modification of model parameters or the upload of simulation data stored in the assistant. Adaptor objects couple technical processes to an IPANEMA application. They organise the D/A- and A/D conversion of control and measured values.

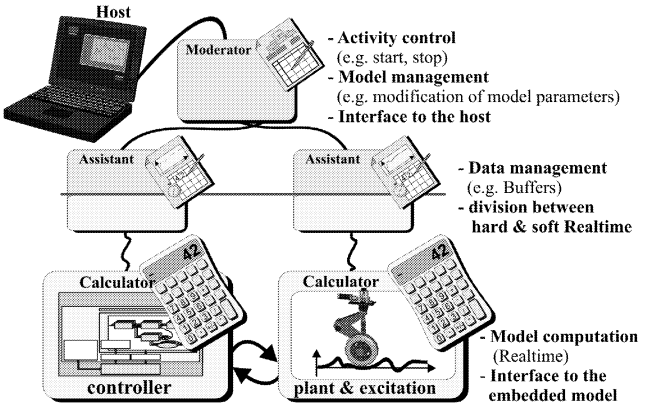


Figure 6. Topology of an IPANEMA application

The topology of the distributed system model defines the structure of an IPANEMA application, with every single submodel being allocated to an IPANEMA calculator. The result is the IPANEMA topology displayed in Fig. 6.

EXECUTION ORDER OF THE DISTRIBUTED SYSTEM MODEL

For a correct execution of the simulation the overall evaluation order has to be taken into account in order to prevent a deadlock during computation of a simulation step that may block the entire simulation (Honekamp 1998).

The connectibility of subsystems has a direct impact on the method of the so-called model integration (Honekamp 1998), i.e., embedding subsystems into the simulation platform. In the process one has to distinguish between different kinds of integration: In a black-box integration, communication can disregard the inner structure of the subsystem while in a white-box integration the communication routines are implemented purposefully into the evaluation functions of the subsystems. The latter case will not be inspected here. Fig. 7 displays the differences between the two procedures:

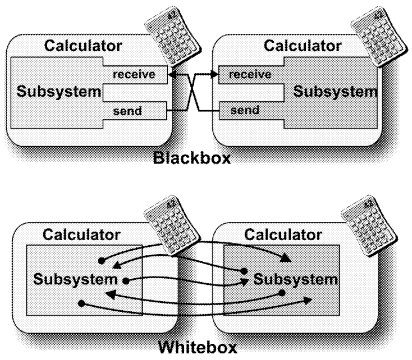


Figure 7. Principle of the black-box- and white-box communications

BLACK-BOX INTEGRATION

In a black-box integration, only the input/output interfaces of the submodels are considered. Their inner structure remains hidden. Therefore communication occurs only vectorially at certain, precisely defined moments in the program run (Honekamp 1998):

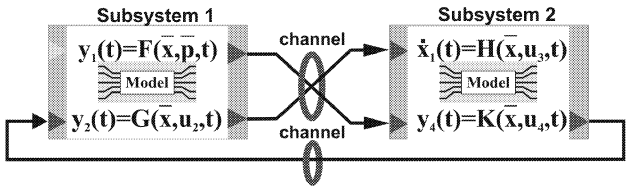


Figure 8. Example: Distributed system with simple feedback and vectorial data transfer

To illustrate this fact, Fig. 8 shows two coupled subsystems resp. submodels, with the entire model comprising a simple feedback.

If the output variables of some computational paths depend directly on one or more input variables, these paths are called direct-link paths because they allow direct access to the result of the computation (Homburg 1993).

If the subsystems communicate among themselves vectorially, a deadlock may occur. Fig. 9 displays the vectorial data transfer, every subsystem executing two computational steps and one communication. In all, this example requires four computational steps and two communications per computing cycle:

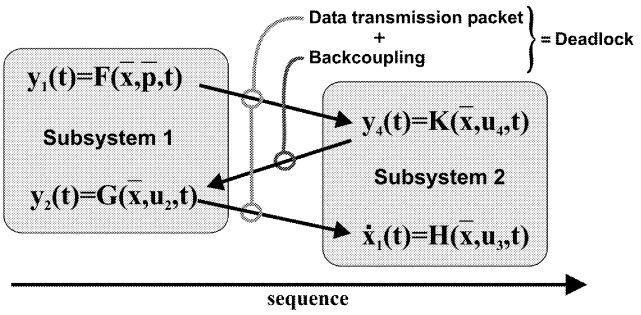


Figure 9. Evaluation of the equations from Fig. 8 with vectorial data transfer

In order to reduce dependencies the equations are sorted according to the categories **non-direct link, direct link,** and **state equations**. These categories serve to classify the input/ output variables (Homburg 1993).

Non-direct links: All output variables that do not depend directly on input values are non-direct links and can thus be computed directly:

$$\underline{y}_N(t) = \underline{F}(\underline{x}, \underline{p}, t) \tag{1}$$

Direct links: This category comprises all input variables that have an immediate effect on at least one output variable:

$$\underline{y}_D(t) = \underline{F}(\underline{x}, \underline{\boldsymbol{u}}, \underline{p}, t) \tag{2}$$

State variables: Input variables that apply only to state equations fall into this category:

$$\dot{x}(t) = \underline{F}(\underline{x}, \underline{\boldsymbol{u}}, \underline{p}, t) \tag{3}$$

Owing to these modifications it is possible to execute deadlock-free communication (or to compute a loop-free evaluation order) with a large number of distributed models. If the decomposition of the equations is applied to the example from Fig. 8 (see Fig. 10), there will be no further deadlock during computation. In the subsystems (see Fig. 10) the ND -, D-, and S-blocks are computed individually and transferred immediately to the connected systems so that four computation steps and three communications are required altogether for a computational cycle:

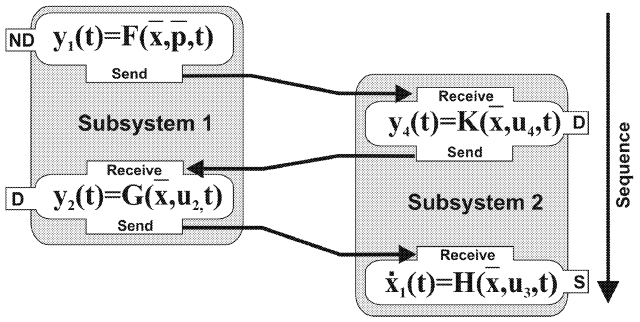


Figure 10. Equations with separate transfer of non-direct-link and direct-link equations from Fig. 8

However, in principle, the problem of a communication deadlock cannot be fully solved just by introducing the ND- , D-, and S-blocks. A deadlock may occur between two subsystems if the couplings between their direct-link blocks make up a loop (Fig. 11). The model cannot be evaluated any more because the subsystems required to compute the data are waiting for the time data of the other subsystems included in the loop and thus are blocked on each other:

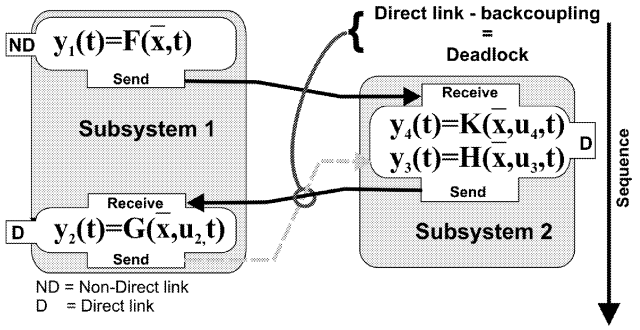


Figure 11. Deadlock due to a loop in the direct-link path

The deadlock can be dissolved in several ways: one can employ iteration (Rükgauer 1996); yet, these do not seem appropriate for real-time processing. A feasible solution is the use of filters (Azizi 1990); however, they alter the model and thus the system behaviour. In most cases, this is inacceptable (Vöcking 2003).

Another solution is to use white-box communication. To this end, communications are required within the D code. A modular code generation is particularly difficult here since the sequence of the evaluation can be computed only for the entire system. This must be considered in the draft of the structure of the modular code. A new approach uses the dynamic computation of the evaluation order at runtime (Oberschelp et al. 2001). Developing a modular code appropriate for a dynamic computation of the evaluation order will be a task of the future.

APPLICATION: SIMULATION OF A COACH WITH AN ACTIVE SUSPENSION

The MLaP boasts of a longstanding tradition in the field of active suspension. It was involved in realizing six test vehicles for various European car manufacturers (Hestermeyer et al. 2001). Also the "0404 Innovisia" coach with active suspension was partly developed by the MLaP; it was the first coach in the world to be equipped with an active suspension (Becker et al. 1996; Becker 2001). An active suspension is superior to a passive one in that it considerably increases ride safety and ride comfort.

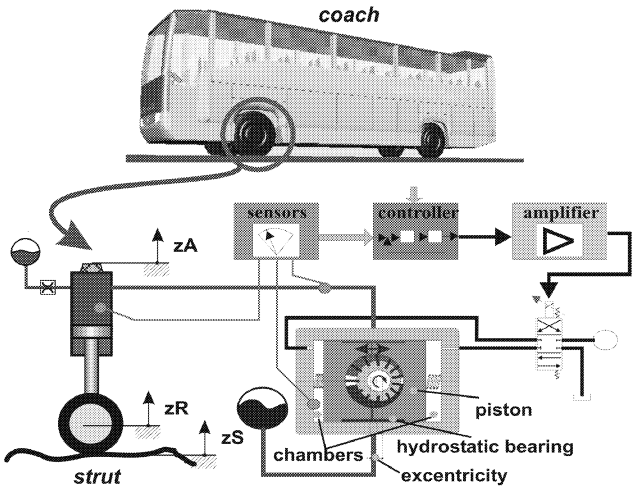


Figure 12. Scheme of active suspension of the coach

MODEL OF THE COACH

The development aims at an optimal design of the control algorithm and the controller parameters in order to provide the system with a controlled motion behaviour. This requires a reduced mechanical model of the coach that represents the essential physical features and parameters:

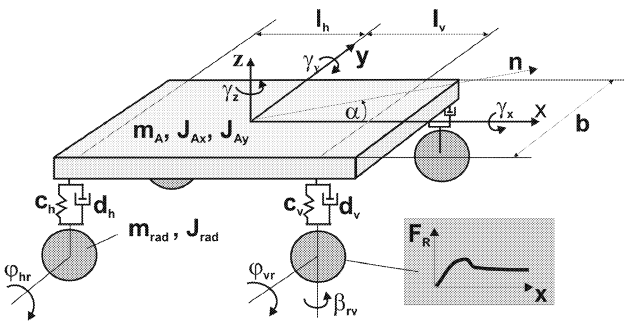


Figure 13. Mechanical substitute model of the coach

Fig. 13 shows a four-wheel vehicle model as a rigid-body system. The vehicle model can execute pitching, rolling, and lifting motions. There is a restriction: the wheel masses can only move in vertical direction. Thus the model has 10 degrees of freedom. Six of them fall to the translatory and the rotatory motion of the coach-body mass and one each to the translatory motion of the wheel masses. In all, the substitute model is described by a system of differential equations of 20th order (Mitschke 1997).

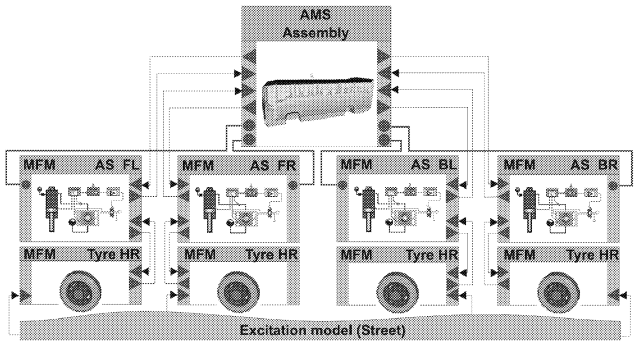


Figure 14. Reduced coach model in simplified representation

Fig. 14 presents the reduced model of the coach including the controller algorithms in simplified representation. The entire model is divided into the coach body, four suspension struts, four wheel modules and one excitation model (the road). The struts and the wheel modules each represent an MFM. The superordinated AMS comprises the coach body in addition to the four MFMs. In all, the model is described by a system of differential equations of 56th order, with 36 states describing the control and the hydraulics. In the following, the model is prepared for modular simulation by a transformation into a distributed system model. In the process, the two front struts are split off from the entire system and make up independent submodels!

SIMULATION RESULTS OF THE MODULAR SIMULATION OF THE COACH MODEL

The correspondence in quality between the modular simulation and the monolithic one is checked by means of a simulation example. The coach-body positions in the coupling points of the struts are measured in relation to the average road conditions and graphically contrasted with one another:

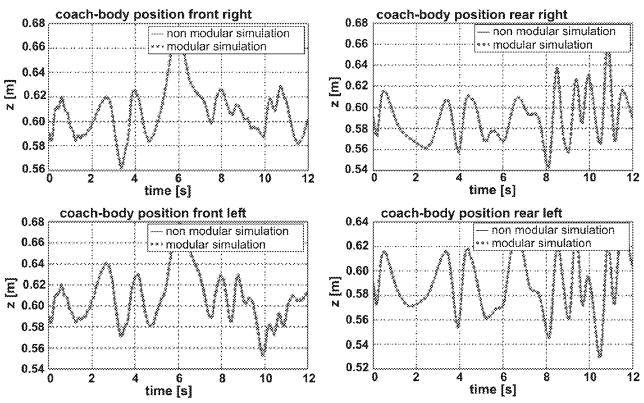


Figure 15. Comparison of the coach-body positions in modular and monolithic simulation

Fig. 15 contrasts the results of the monolithic simulation with those of the modular one. In either case a 4th order Runge-Kutta method with a stepsize of 0.001 sec was applied. Off-line simulations were performed on a Windows PC.

With both these simulations the graphs are identical. This was yet to be expected because the mathematics of the model is not altered in the decomposition process but evaluated by means of the same procedures. In this particular testing routine, deviating simulation graphs would point to errors in the concept and implementation, e.g., errors in communication or in the three-step evaluation.

SUMMARY AND OUTLOOK

We presented the concept of modular-hierarchical decomposition and distributed simulation of mechatronic system models. The concept comprises a structuring of mechatronic systems, a decomposition of the model, and a coupling of the IPANEMA runtime platform. Special emphasis was laid on realization in the CAMeL-View modelling tool for the modular-hierarchical decomposition of models. This considerably simplifies the consistently structured design of mechatronic systems from modelling to distributed off-line simulation to distributed hardware-in-the-loop simulation. As an example of an application the modular off-line simulation of a coach model with active suspension was presented.

Future work on this subject matter is bound to focus on two main directions: Firstly, realization of distributed applications on distributed, heterogeneous real-time networks has to become simpler for the user. This goal can be reached by a model description of the target hardware. Here, structure and configuration of the hardware are described; additionally, the description comprises the mapping of the subsystems to the hardware modules. On the basis of this overall description, an automatical generation of an IPANEMA application tailored to the hardware should be possible. Another important aspect of our research centers on modifications to the code generation and to the IPANEMA runtime platform in view of applications with a structure that can be altered during runtime.

At present, extensive research work is done on this in the context of the Collaborative Research Center 614 "Self-Optimizing Concepts and Structures in Mechanical Engineering" and is published on its behalf and funded by the Deutsche Forschungsgemeinschaft.

REFERENCES

Azizi, S. A. 1990. *Entwurf und Realisierung digitaler Filter.* 5. Aufl., Oldenbourg-Verlag, München/Wien.

Becker, M.; Jäker, K.-P.; Frühauf, F. and Rutz, R. 1996. "Development of an Active Suspension System for a Mercedes-Benz Coach (O404)". In: P*roceedings of the 1996 IEEE International Symposium on Computer-Aided Control System Design*, Dearborn, MI.

Becker, M. 2001. "Eine flexible, kompakte, hydraulische Antriebseinheit für die aktive Federung". In: *Proceedings of VDI-Mechatronik-Tagung*, Frankenthal.

Ditze, C. 1998. "A Step towards Operating System Synthesis". In: *Proceedings of the 5th Annual Australasian Conf. on Parallel And Real-Time Systems (PART) IFIP.* IEEE, Adelaide.

dSPACE GmbH. 1995. *Real-Time Interface to SIMULINK.* User's Guide, Paderborn.

ETAS GmbH. 1996. *ASCETS-SD: User Guide Version 1.0.* Schwieberdingen.

Gausemeier, J. and Lückel, J. 2000. *Entwicklungsumgebung Mechatronik.* HNI-Verlagsschriftenreihe, Bd. 80, Paderborn.

Gambuzza, A. 2002. *Konzept zur verteilten modularen Simulation mechatronischer Systeme,* Diplomarbeit, Universität Paderborn.

Hahn, M. and Koch, T. 2000. "CAMeL-View - Ein Werkzeug zum integrierten CAD-gestützten Entwurf mechatronischer Systeme". In: *Proceedings of "Simulation im Maschinenbau" (SIM2000)*, Dresden.

Heimann, B.; Gerth, W. and Popp, K. 2001. *Mechatronik: Komponenten, Methoden, Beispiele.* Fachbuchverlag Leipzig im Carl Hanser Verlag, München/Wien.

Hestermeyer, T.; Becker, M. and Neuendorf, N. 2001. "Nichtlineare ABC-Regelungen mit Operator-Controller-Struktur, abgestimmt auf Führung und Störung der Straße". In: *Proceedings of Haus der Technik: Driveability*, Essen.

Homburg, C. 1993. *Entwurf und Implementierung einer standardisierten Beschreibungsform mechatronischer Systeme*, Diplomarbeit, Universität Paderborn.

Honekamp, U. 1998. *IPANEMA - Verteilte Echtzeit-Informationsverarbeitung in mechatronischen Systemen.* Diss., Universität Paderborn; VDI-Verlag, Düsseldorf.

Lückel, J.; Hestermeyer, T. and Liu-Henke, X. 2001. "Generalization of the Cascade Principle in View of a Structured Form of Mechatronic Systems". In: *Proceedings of 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics AIM 2001*, Villa Olmo, Como.

Kiffmeier, U. 1995. *Multi-DSP-Power auf Knopfdruck.* Elektronik 12/1995.

Mitschke, M. 1997. *Dynamik der Kraftfahrzeuge, Band B: Schwingungen.* 3. Auflage, Springer-Verlag, Berlin/Heidelberg/New York.

Oberschelp, O.; Homburg, C.; Deppe, M.; Gambuzza, A. and Seuss, J. 2001. "Verarbeitungsorientierte Darstellung verteilter hybrider Systeme der Mechatronik". In: *Proceedings of 5. Magdeburger Maschinenbautage*, Otto-von-Guericke-Universität, Magdeburg.

Rükgauer, A. 1997. *Modulare Simulation mechatronischer Systeme mit Anwendung in der Fahrzeugdynamik.* Diss., Institut B für Mechanik, Universität Stuttgart; VDI-Verlag, Düsseldorf.

Schumacher, M. A. I. 2002. "OSEKtime - ein zeitgesteuertes Standard-Betriebssystem". In: *Elektronik Automotive 02/2002.*

Vöcking, H. 2003. *Multirate-Verfahren und Umschaltstrategien für verteilte Reglersysteme,* Diplomarbeit, Universität Paderborn.

Zanella, M. C. and Stolpe, R. 1998. "Distributed HIL Simulation of Mechatronic Systems Applied to an Agricultural Machine". In *Proceedings of DIPES '98*, Schloß Eringerfeld.

A Simulation Study for Assessing the Performance of Manufacturing System of Aerospace Industry

Mohd Kamal Mohd Nawawi
Razman Mat Tahar
School of Quantitative Sciences
University Utara Malaysia, 06010
Sintok, Kedah, MALAYSIA

1. Introduction

Simulation has been successfully applied in many different areas such as manufacturing, service system, healthcare, transportation, supply chain etc (Harrel and Tumay, 1995). The opportunities to cut costs and to improve service levels in these sectors are tremendous by applying this technology. The design of new manufacturing systems or improving the existing system can be immensely supported by simulation as the designer is given an opportunity to assess the proposed system via properly designed experiments without the cost and time associated with physically building the system. This study focuses on applying a simulation technology to a manufacturing system of an aerospace industry in Malaysia. The manufacturing plant under study is a joint venture between the two Malaysian companies and two other American companies. The plant involves in the production of advanced composites materials, i.e. producing fixed trailing edge parts for the wings of an aircraft.

2. System Description

Figure 1 shows the general flow of the production processes at the plant under study. Current production rate of the plant is 110 parts per day.

In this study, the process begin when a semi-finish product flow in a single part order along moving lines in lay-up cells through autoclave, water-jet trim, test, paint, inspection and finally to shipping. Batch processing is also applied at a few workstations.

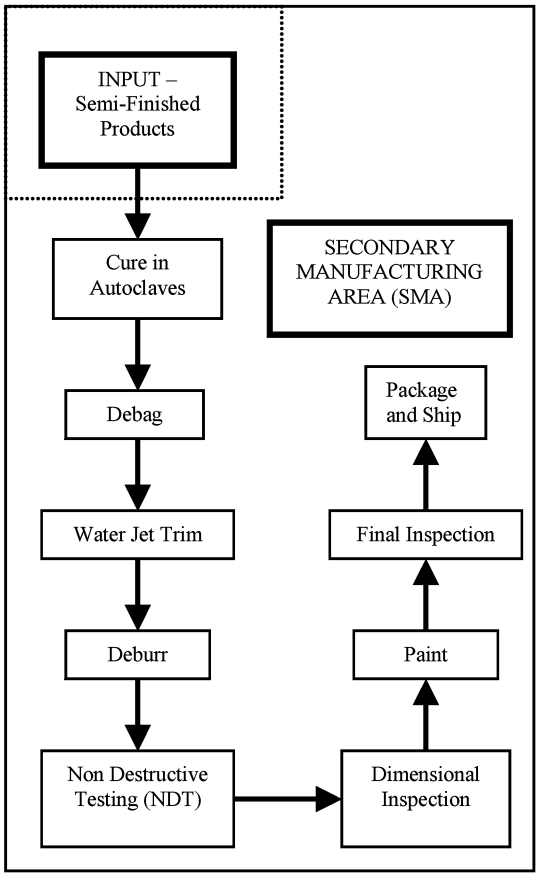


Figure 1. General Flow of Production Process

3. Goal of Study

Simulation study at this plant is initiated because the workload for an existing system is predicted to change. The managers need to understand the impact of such changes on system performance. The simulation model offers a practical information to the management to make

informed decision on performance of this system.
The main objective of the study is to model and simulate the design and operational policies of the production process, which can be used to improve the performance of the different activities at the plant. The study also intends to evaluate the performance of existing operations and capability of the plant resources. It can then be used to find an alternative design of optimum production system

The study was to provide information on machine utilization, part flow time, and information on bottlenecks. The simulation was also used to test "what if " scenarios such as increased production requirements, resources utilization and other effects of operating variables on production.

4. Model Development

The first step of the project was to define and formulate the problem. Understanding the problem clearly will make the modelling task become much easier. The second step was the model development. The model development is the most visible part of the simulation study. The model development will adhere to the goals and objectives and will be completed in phases of increasing complexity. The model will first capture the basic logic of the system and the flow logic. Part movement and elements will be added and verified as the model is developed. As soon as basic model function has been encoded, more detail can be added for each location until the desired function is achieved. The last step was doing the experimentation and presenting the analysis of results. The software chosen for this case study was ARENA by Rockwell Software. It is a menu driven package and run on an IBM compatible PC. The resulting layout was animated when the simulation was run, showing the movement of parts and resources with elapsed time. The simulation can be interrupted at any stage and a comprehensive reporting system can be viewed.

4.1 Input Data

Data analysis provides the driving force for any simulation model. Without input data, the simulation model itself is incapable of generating any data about the behavior of the system it represents (Bank and Carson, 1990). Input data were collected from the production plant and a comprehensive real life data on plant operation analyses was carried out. The distributions of process times at eight stations were fitted, using 50 samples of actual data (for each of the process except for dimensional inspection process with 100 samples of data).

Process	Distribution	Expression
Debag	Triangular	TRIA(3.07, 3.52, 3.98)
Water Jet Trim	Beta	4.9 + 1.04 * BETA(1.83, 2.38)
Deburr	Beta	15.1 + 4.75 * BETA(1.87, 2.11)
NDT	Normal	NORM(90, 0.784)
Dimensional Inspection	Weibull	10 + WEIB(5.27, 2.6)
Rework	Uniform	UNIF(40, 50)
Paint	Beta	7 + 1.89 * BETA(1.86, 2.28)
Final Inspection	Triangular	TRIA(8.01, 8.49, 8.98)

Table 2. Distribution of process times

5. Simulation Results

The statistics collected from the simulation model include parts throughput, parts flow times, and work-in-process quantity

(WIP). The simulation model was run for 5 replications and the average were recorded.

5.1 Parts throughput

Throughput represents the number of parts for the period of one-week study. Table 3 shows the output of the throughput using simulation compared with the actual plant data and it seem that they are in good agreement. The extreme case of the difference is only 3.22%.

Part Type	Average Simulation	Historical Data	% Difference
Type A	96.8	99.4	2.62
Type B	117.8	118.8	0.84
Type C	115	118.8	3.20
Type D	96.2	99.4	3.22

Table 3. Number of predicted parts compared with historical data.

5.2 Parts flow time

This is the total time that a part spends in the system to complete all the activities from cutting to packing.

Part Type	Average Simulation (hr)	Actual Data (hr)	% Difference
Type A	28.2961	30	5.68
Type B	27.8858	30	7.05
Type C	27.7921	30	7.36
Type D	28.8001	30	4.00

Table 4. Parts Flow Time

The flow time should be kept to a minimum to reduce work-in-process inventories which carry hidden cost. The average parts flow time is compared with the actual data together with calculated error is given in Table 4.

5.3 Work-In-Process (WIP)

The WIP for each type of part were also computed from the model. The output from the simulation model is then compared with the historical data. The results are shown in Table 5.

Part Type	Average Simulation(min)
Type A	44.92
Type B	52.96
Type C	52.88
Type D	45.77
Total	**196.54**
Actual Data	**197**
% Difference	**0.23**

Table 5. WIP for each type of parts.

6. Model Experimentation

In line with the plan of company to expand its business by producing more parts, an increase of 20% of production capacity was experimented. The number of daily arrival was increased from 110 parts to 132 parts to investigate whether the present resource capacity can tolerate. Besides, additional two alternative scenarios were also experimented. One of the scenarios is by an increase of water jet trim machine operation time by 2 hours (20% increase) and 1.5 hours (18.75% increase) extra time for other resources. The other scenario is by an increase of 2 hours extra time for resource with high utilisation (>95%) i.e. deburr operators, final inspectors, NDT machine and waterjet machine, and 1.5 hour for other resources (ply cutters, layup operators, painters, dimensional inspectors, rework operator, and debag operator). The simulation model was run for 5 replications and the average was recorded for each of the scenarios.

6.1 Parts throughput

Table 6 shows and explains that present resource capacity (Scenario 1) was unable to cater the production increase.

However, Scenario 2 and 3 seem that they are in good agreement with 20% expected increase of throughput.

Production throughput	Simulation Output	Sc 1	Sc 2	Sc 3
Type A	96.8	103.6	117.4	117.2
Type B	117.8	119.2	140	141.4
Type C	115	120.2	141.6	140
Type D	96.2	99.8	119.4	118.2
Total Throughput	425.8	442.8	518.4	516.8
Difference to simulation output		4%	22%	21%

Table 6. Number of predicted parts throughput

6.2 Parts flow time

Parts flow time of the three scenarios is given in Table 7. Scenario 1 shows that the parts will spend about 30% extra time compared to other scenarios, where the parts will spend less time in the system.

Flow time	Simulation Output	Sc 1 (% diff)	Sc 2 (% diff)	Sc 3 (% diff)
Type A	28.2961	36.594 (29%)	26.0743 (-8%)	26.174 (-7%)
Type B	27.8858	37.4852 (34%)	26.6088 (-5%)	26.0064 (-7%)
Type C	27.7921	37.3099 (34%)	26.2804 (-5%)	25.7439 (-7%)
Type D	28.8001	37.516 (30%)	26.6815 (-7%)	26.6476 (-7%)

Table 7. Flow time of parts (hours)

6.3 Work-In-Process (WIP)

Parts WIP of the three scenarios is given in Table 8. Scenario 1 shows that WIP value is much higher compared to other scenarios.

Work-in-process (WIP)	Simulation Output	Sc 1 (% diff)	Sc 2 (% diff)	Sc 3 (% diff)
Plies Type A	44.92	69.4871 (55%)	49.5122 (10%)	49.92 (11%)
Plies Type B	52.96	85.8284 (62%)	60.8769 (15%)	59.37 (12%)
Plies Type C	52.88	85.5136 (62%)	60.0425 (14%)	58.91 (11%)
Plies Type D	45.77	71.8447 (57%)	50.4458 (10%)	50.72 (11%)
Total	196.53	312.674 (59%)	220.877 (12%)	218.91 (11%)

Table 8. WIP for each of scenario

7. Conclusion

This paper presents the results of a case study, which involved the use of computer simulation technique for production planning process of aerospace industry. The model built is used to investigate a variety of issues, for example to determine the impact of a proposed change, without affecting production. The model is also able to determine the plant capacity under various situations. This enhances the ability to manage the system, control its capacity, and make better decisions regarding its operation, which in turn improves ability to deliver quality product to customers.

When production rate was increased by 20% to investigate the current plant capacity, the current resources capacity was unable to tolerate with this increment as experimented in Scenario 1. Obviously, the parts flow time and WIP increase tremendously with the 20% increase of parts demand. Scenario 2 deals with the 20% production rate increase with an increase of water jet trim machine operation time by 2 hours (20% increase) and 1.5 hours (18.75% increase) extra time for other resources. However, the utilisation of deburr operators, final inspectors, NDT machine and waterjet machine found to be very high.

The research and the simulation model developed have improved understanding of the inter-relationship of the several physical components of the plant. The process of constructing the simulation models and reviewing the interaction of these components has given an insight into the different operational characteristics at the plant. The approach of system analysis is not only beneficial to the modeler, but it is also useful to the planner since it gives a thorough understanding how the plant behaves and not how one thinks it behaves.

REFERENCES

1. Banks, J., J.S. Carson II, B.L. Nelson, and D.M. Nicol, 2000 Discrete Event System Simulation, 3rd Ed., Prentice-Hall.
2. Carrie, A., 1988, Simulation of Manufacturing Systems, John Wiley & Sons, New York, NY.
3. Harrel, C.R. and K.Tumay, (1995) " Simulation Made Easy, A Manager's Guide". Institute of Engineers, Narcross, Georgia.
4. Law, A.M. and W.D. Kelton, 2000. Simulation Modeling and Analysis, 3rd Ed., McGraw-Hill.
5. Maria, A., (1997), Introduction To Modeling And Simulation, Proceedings of the 1997 Winter Simulation Conference. Pg 7-13. ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson.
6. Miller, S., D. Pegden (2000), Introduction to Manufacturing Simulation, Proceedings of the 2000 Winter Simulation Conference pg 63-66.
7. Shannon, R.E., (1998), Introduction to The Art and Science of Simulation, Proceedings of the 1998 Winter Simulation Conference. Pg 7-14.

VALIDATION, ANALYSIS AND EVALUATION

SIMULATION SOFTWARE AND MATHEMATICAL ANALYSIS OF SIMULATION RESULTS

Johann Christoph Strelen
Rheinische Friedrich–Wilhelms–Universität Bonn
Römerstr. 164, 53117 Bonn, Germany
E-mail: strelen@cs.uni-bonn.de

KEYWORDS

Discrete Simulation, Software, Statistical Analysis

ABSTRACT

Software packages for discrete simulation which do not support statistically correct results entail methodically incorrect simulation models which may lead to wrong conclusions based on unreliable figures. However, it is not difficult to create better tools. We are going to present some simple ideas how this can be obtained with quite straightforward techniques which are easy to apply in simulation and easy to implement in simulation software packages. Some wide-spread packages do not offer comfortable support for controlling the length of simulation runs depending on intermediate statistical evaluation, although this is crucial for sound simulation studies.

INTRODUCTION

Discrete-event stochastic simulation is a very important technique for evaluating performance and reliability. It is used in many application areas. Due to the stochastic nature of the results, careful statistic analysis must be done for the correct interpretation of calculated values. If this is omitted, there is a significant probability of making erroneous inferences about the system under study.

"As a matter of fact, a very common mode of operation is to make a single simulation run of somewhat arbitrary length and then to treat the resulting simulation estimates as the "true" model characteristics. Since random samples from probability distributions are typically used to introduce the influence from outside of the model, the estimates are just particular realisations of random variables that may have large variances. As a result, the estimates could, in a particular simulation run, differ a great deal from the corresponding true values of the model" ([7]).

There are many special-purpose simulation packages available which provide comfortable means for programming simulation models on a high level of abstraction, resulting in a significant decrease of programming time and in a reduction in overall project costs.

As a matter of fact, many simulation tools do not provide sufficient support for obtaining correct statistical analysis of results. Probably, such a calculation can be implemented using the implementation language of the package, e.g. Siman in the package Arena [6]. But this is quite cumbersome, for the modeller must learn a second user interface. Sometimes, a script language must be used for controlling replications of simulation runs with different random numbers. In this case, the modeller would have to know a third language.

High-level constructs of a simulation language usually have the advantage that details, for example how confidence intervals are calculated, remain hidden from the programmer. This helps to avoid programming bugs. On the other hand, it is obvious that the modeller needs to know in detail what happens inside his model.

If he wishes to learn about details, he has to refer to printed or online help which quite often does not provide clear and precise information and is written badly. Much time is wasted on searching for the required information.

Moreover, comfortable packages may be inefficient in some respect. Sometimes doing many replications is costly or even impossible, or millions of data, most of it unimportant, must be collected in files in order to obtain only a few important figures which afterwards must be selected for the mathematical analysis.

Some desirable features of simulation models are even impossible to implement, for example the continuation of all simulation runs of an experiment after they were stopped. But this may be important for obtaining smaller confidence intervals of controlled width.

For example, if a modeller uses the simulation package OMNET++ [1], he programs in C++, builds the model configuration with NED, OMNETs net description language, controls the simulation with a platform-dependent script language, must learn some file formats for initialization and configuration, and two more file formats for the result files. These files contain huge quantities of result data which afterwards have to be re-filtered by a special program. Than the mathematical analysis has to be programmed. There is no support provided for sequential procedures.

This paper neither presents a novel modelling technique, nor does it deliver a solution for a long-known problem.

We hope, however, that it will inspire modellers and tool designers to reconsider this problem for which solutions do exist which too often are not applied. We hope that simulation experts will realize that it is not even that difficult to cope with this problem and improve on things.

In short, our message is as follows:

1. Many simulation studies are methodically not correct with respect to statistical analysis of the results, many wrong conclusions are drawn based on unreliable figures.

2. Simulation software packages which do not support statistically correct results entail this.

3. It is not difficult to make better tools, better in this respect.

4. We indicate some simple ideas how this can be obtained with quite straightforward techniques which are easy to apply in simulation and easy to implement in simulation software packages.

In our view, many modellers and tool designers are not aware of the importance of correct statistical inference or are not even interested in it. We are afraid that it is very difficult, nearly impossible, to overcome this unawareness and prejudice.

However, we are convinced that parts of this problem are based on a simple misunderstanding that can easily be removed, namely the opinion that it is very difficult to achieve improvements.

Clearly, statistics is a hard discipline, but it is quite easy to understand which are the crucial aspects with simulation output data, and it is easy to apply some simple techniques which are much more correct and reliable than the usual "perform one simulation run and take the obtained figures as the true solution". The key is that simulation tools must provide these techniques and the according control of simulation runs. We present these demands on the tools in the sequel.

Desirable simulation software properties concern the process of modelling and programming a simulation model, and the model itself. Here we are interested in the correct, solid analysis of simulation results and in the ease of programming it. To this end, a tool must offer the statistical algorithms themselves, some specific modes of simulation run execution, an algorithm for detecting the end of the transient phase, some features of random number and random variate generation, some features which concern the collecting of statistical data, and some general support.

1 CONFIDENCE INTERVALS

We suggest that simulation software should implement three methods of calculating confidence intervals:

- The batch-means method
- The replication/deletion method
- Median confidence intervals

There are other, more elaborated techniques for confidence intervals, such as the regenerative method, autoregressive processes, the spectral estimation method, or the standardized time series method, see Fishman [5], Bratley, Fox, and Schrage [4], Banks [3], or Law and Kelton [7]. A specific one of these techniques may apply to one model but not to another. The different assumptions must be checked. The modeller who wants to use it must know and understand it to some degree.

The three above techniques are general and are easy to apply without knowledge of statistics, and compared to the more elaborated techniques the preconditions for achieving statistically correct results are better fulfilled. Due to this, these simple confidence intervals are often more accurate, see the empirical comparisons in [7] and [8, 9], since in simulation, the assumptions are never fulfilled literally, only nearly, more or less. Accurate here means the following: The probability that a confidence interval contains the true value equals the confidence level. In the studies, confidence intervals are often too optimistic, e.g. the coverage probability is only 70% while the assumed confidence level is 90%.

Median confidence intervals are new and unknown, but for simulation results they have some favourable features. Therefore we summarize them in section 7.

The replication/deletion approach and the batch-means method use classical confidence intervals. Given a sample $(Y_1, ..., Y_n)$ of independent, normally distributed random variables, a confidence interval with confidence level $1 - \alpha$, $0 < \alpha < 1$, is $\bar{Y} \pm t_{n-1,1-\alpha/2}\sqrt{S2/n}$ where $\bar{Y} = (Y_1 + ... + Y_n)/n$ is the sample mean, $S2 = (Y_1^2 + ... + Y_n^2)/(n-1) - \frac{n}{n-1}\bar{Y}^2$ the sample variance, and $t_{n-1,1-\alpha/2}$ the $(1 - \alpha/2)$-quantile of the Student distribution with $n - 1$ degrees of freedom. If $n > 120$, the $(1 - \alpha/2)$-quantile $u_{1-\alpha/2}$ of the standard normal distribution can be taken instead which does not depend on n. In [2] one finds approximations for these inverse distribution functions, t and u, or simply a table with the quantiles is stored for interesting confidence levels like 90%, 95%, 99%, 99.9% and $n = 1, ..., 120$ and $n > 120$. Our strange formula for the sample variance indicates that not all values Y_i must be stored, only the two sums $Y_1 + ... + Y_i$ and $Y_1^2 + ... + Y_i^2$ must be updated with each new value Y_{i+1}.

The median confidence interval is $[\min_{1 \le i \le n} Y_i, \max_{1 \le i \le n} Y_i]$ where the confidence level is a function of n, $1 - \alpha = 1 - 1/2^{n-1}$. Hence there is only a limited amount of possible confidence levels, e.g. 87.5%, 93.75%, 96.9%, 98.4%, 99.2%,...,99.9%,... ; n must be chosen accordingly. This is not a significant restriction.

The key for reliable simulation results is the availability of high quantities of statistically independent data.

Usually, the modeller desires confidence intervals of some given maximum width. He does some simulation, calculates a confidence interval, and if it is too wide, he

does some more simulation, recalculates, and so on, until the confidence interval is small enough.

There are two types of simulation: Terminating and nonterminating.

A terminating simulation run starts in a specific given model state and ends at a defined event. A single value or a mean is stored for each model parameter under examination. For confidence intervals, some or many runs (replications) with different random numbers are performed. This is the replication/deletion method. Terminating simulations are suitable for nonstationary (non-ergodic) models.

A nonterminating simulation run consists in a transient phase in which no data are collected, and a subsequent steady-state phase where data are useful. Usually the gathered observations $X_1, X_2, ...$ of a specific model parameter are dependent, whereas for our preferred simple confidence interval methods we require independent, or at least approximately independent, data. This is achieved with batch-means or independent replications. An algorithm which extends a simulation to the extend a (median) confidence interval is too wide is called a sequential procedure. In literature, ingenious sequential procedures are proposed which are based on batch-means, independent replications, regenerative points, or spectrum analysis. Here again, we confine ourselves to batch-means or independent replications.

1.1 Batch Means Method

For this method, the observations of a nonterminating simulation run are grouped into n batches, each with l observations, and for each batch i, $i = 1, ..., n$, the batch mean $\bar{X}_i = (X_{(i-1)l+1}, ..., X_{il})/l$ is taken as datum Y_i for the (median) confidence interval. This is justified by the observation that the longer the batches are, i.e. the larger l is, the more statistically independent the batch means will be.

A sequential procedure may extend the simulation run and add additional batches or augment their length l. However, it is not recommendable to consider more batches, because few longer batches have better statistical properties than many short ones - they are statistically more independent, and the crucial normality assumption for the estimators is fulfilled to a higher degree. Moreover, for median confidence intervals, the choice of n is crucial, it defines the confidence level. Therefore it is important for the batch means method that the batches can be dynamically extended - this is a demand concerning the organization of simulation.

Now, if a (median) confidence interval has the half length h, but should have the shorter half length γh, $0 < \gamma < 1$, then l/γ^2 is a good guess for the longer batch length, the batches should be extended by the factor $1/\gamma^2$. This can be realized without storing extra data if the factor, say f, is chosen to be integer, e.g. $f = \lceil 1/\gamma^2 \rceil$: The batch means $\bar{X}_i$ and newly simulated data $X_{nl+1}, ..., X_{nfl}$ are used for the new batch means

$$\bar{X}_i^{\text{new}} = \begin{cases} (\bar{X}_{(i-1)f+1} + ... + \bar{X}_{if})/f \\ \quad \text{if } if \le n, \\ \left[(\bar{X}_{(i-1)f+1} + ... + \bar{X}_n)l + \right. \\ \quad \left. X_{nl+1} + ... + X_{ifl}\right]/(fl) \\ \quad \text{if } (i-1)f < n < if, \\ (X_{(i-1)fl+1} + ... + X_{ifl})/(fl) \\ \quad \text{otherwise,} \end{cases}$$

$$i = 1, ..., n.$$

For this mode of operation, a simulation tool must allow intermediate calculation of confidence intervals and continuation of the simulation run depending on their lengths.

1.2 Replication/Deletion Method

When model parameters are to be estimated by terminating simulation, in some independent replications realisations are calculated. We consider realisation X_j in replication j for a specific parameter. X_j may be a single value, e.g. the number of served items in the whole simulation run, or it can be a mean, e.g. the mean of all waiting times of the items. Since the X_j are independent, confidence intervals can be calculated immediately. If the confidence interval is too wide, additional replications are performed. Here again, with the factor f for the number of replications the length should be nearly reduced to the wished-for extent.

For this mode of operation, a tool must support efficiently many replications.

It is not recommended to use the values X_j of many replications directly, they should be grouped into n groups where n is between 5 and 10 for classical confidence intervals, or is determined by the confidence level of a median confidence interval. From the values X_j of each group i, their mean $\bar{X}_i$ is used as value Y_i in the classical confidence interval formula, as above with batch means. In general, the assuptions for the confidence interval method are better fulfilled by these means.

For nonterminating simulations with the replication/deletion technique, it is important that all replications of a simulation experiment can be extended depending on intermediate results.

Each replication begins with the transient phase. First, its length must be determined. This is achieved by simulating a predefined number of independent replications and continuing each replication until stationary conditions are observed, e.g. with the technique of Welsh [7, 10].

Afterwards, the simulation is continued in each replication until the confidence intervals are small enough. To this end, from time to time, the confidence intervals are calculated, and depending on their length, the simulation is continued or not. Here again, the factor f applies. The (median) confidence intervals are calculated with mean $\bar{X}_i$ of the values X_j of the replication i as the values Y_i of the classical confidence interval formula.

Here again, using many short replications is unfavourable because in each of them computing time is be-

ing wasted on the unproductive transient phase. Even with today's fast working computers, computing time is still valuable for simulations, since some simulations require much of it (and some simulation can not be done at all due to extensive computing time).

2 GATHERING DATA

It must be possible to specify which data are collected. Otherwise, resources will be wasted, and it may be cumbersome to filter interesting data out of a huge file with much garbage.

The library of the package should provide comfortable functions for gathering data (statistical counters) and for statistical analysis. Two types of counters are used: For continuous-time statistics like the utilization of a server, weighted sums of states are stored for time averages.

For discrete-time statistics like a mean waiting time, sums of observed realisations are stored, together with their minimum, their maximum, the sum of the squared realisations for the sample variance, histogram data etc. From these values, an according function of the counter can calculate (median) confidence intervals if the summed realisations are at least nearly independent, e.g. means from independent replications or from batches.

It must be possible to initialize statistical counters or to save their contents for later addition of further data.

The same statistical counter must be usable in different simulation runs, for the replication/deletion method, for the antithetic variance method, and for the comparison of different model alternatives.

3 MODES OF SIMULATION RUN EXECUTION

In the discussion about confidence intervals, sequential procedures, and detection of the end of the transient phase, we saw some demands concerning the control of simulation runs:

1. The control of run lengths uses numerical values from simulation results.

2. Depending on such calculated values, a run may be continued or terminated.

3. Some simulation runs (replications) may coexist. They all may be continued or terminated, depending on calculated values.

4. Many - thousands of replications must be supported to be executed efficiently.

Object-oriented languages like C++ support these modes of operation since many objects may exist simultaneously, hence some instances of the model and of the simulator may coexist.

4 RANDOM NUMBERS AND RANDOM VARIATES

For variance reduction, random numbers must be synchronized. That means, the same realisation of a random variable of the model is obtained with exactly the same random number in different simulation runs, even if the model differs in some aspect, for example with an other queueing discipline, a different number of servers, or other. To this end, many streams of random numbers must be available. The modeller must know how long the streams are, and in which stream j the first random number follows the last random number of stream i. If he does, the modeller can use streams of double, triple,... length.

It must be possible to store the actual internal value of a stream with which the stream can be continued later at a defined point.

The modeller must know how random variates are calculated from random numbers which have a uniform distribution between 0 and 1, and he must be able to specify that this is accomplished with the inverse transformation method (where suitable). This is important for antithetic variance reduction and when different models are compared by means of confidence intervals for differences, see [7]. These techniques can be supported in tools without difficulty.

The library of the package should provide the usual random variate generation functions, but user-defined variates must be possible as well, and their usage must not be different from predefined variates.

5 INITIAL STATE AND TRANSIENT PHASE

The library of the package should provide at least one function for the detection of the transient phase in a steady-state simulation.

It must be possible to initialize the state of the model as desired. This can shorten the transient phase in steady-state simulations, and in terminating simulations the initial state is an essential part of the model.

6 GENERAL SUPPORT

6.1 Modeling Language

The modeller who uses a software package should need at most two languages. If the implementation language is a common general purpose language like C++, the user probably knows it and does not have to learn a new language. Without further preparation he can read the implementation program which is the most precise documentation, and can immediately extend the library.

Object-oriented languages like C++ allow to have some simulation model objects coexisting. Thus some simulation runs can be extended when needed. We saw that this

is useful for detecting the end of the transient phase and for sequential procedures.

If a tool designer uses an own implementation language, the same terminology and the same syntax as in common general purpose languages should apply. Exotic inventions of occasional language designers are usually neither ingenious nor esthetical nor practical, but always they create unnecessary pains for the programmer.

It makes sense to have a second language for the specification of a network of model components with their connections. Common general purpose languages are not particularly suitable for the description of networks. Alternatively or additionally, these nets can be specified conveniently with a graphical interface by drag-and-drop.

6.2 Model Documentation

If the model is built with a graphical interface whose components are described in detail dialog boxes, there should be a single command for storing a complete readable description of the whole model as documentation which also can be printed for proof-reading, verification of the program, validation of the model, publishing.

6.3 Package Documentation and Online Help

The professional user of a simulation package knows modelling and statistical analysis of results, but probably he does not know this special package. The documentation must help him to find out how the model and the statistical methods can be implemented. To this end, the index of topics must contain the problem oriented usual terms like customer, waiting queue, customer class, server, serving discipline, routing, arrival process, batch arrivals, random number, variate, Erlang distribution, confidence interval, batch means, replication/deletion method, antithetic variance reduction, transient phase detection etc.

7 MEDIAN CONFIDENCE INTERVALS

Confidence intervals around the median of estimators are a new technique for calculating confidence intervals ([8, 9]). They are a substitute for confidence intervals for the expectation. This is adequate since for many estimators the median and the expectation are close together, or even coincide, particularly if the sample size is large.

Median confidence intervals are easy to obtain, they are given immediately by the smallest and the largest mean in independent replications or from batch means.

The variance of the estimator is not used, even may not exist. Hence, in some cases, median confidence intervals exist where classical confidence intervals do not.

The assumption concerning the distribution of the estimator is only symmetry whereas it is normality for classical confidence intervals. The last assumption implies the first, is clearly a stronger requirement. Hence, in some cases, the assumptions for median confidence intervals are fulfilled whereas they are not for classical confidence intervals.

In simulation studies we found out that median confidence intervals seem to be more accurate than classical confidence intervals: Their coverage probability is closer to the assumed confidence level.

They can be used for estimators where classical confidence intervals are difficult to calculate or are inaccurate, they apply to functions of estimators. In our empirical study, we oserved higher accuracy than with Jackknife confidence intervals.

The confidence level of a median confidence interval is a function of n, the number of independent estimations, namely $1-\alpha = 1-1/2^{n-1}$. That means, only certain confidence levels are possible, n must be chosen accordingly, which is no severe restriction.

CONCLUSION

We showed how simulation software can provide better support to the statistical analysis of simulation results. This is not difficult, no elaborate mathematics must be applied, but the means for controlling simulation run lengths dependent on intermediate results must be present in the tool. We are in the process of preparing a small C++ library with these features as a precise prototype for this kind of simulation software.

References

[1] *http://www.hit.bme.hu/phd/vargaa/omnetpp.htm, OMNET++.*

[2] M. Abramowitz and I.A. Stegun. *Handbook of Mathematical Functions.* Dover Publications, Inc., New York, 1968.

[3] J. Banks. *Handbook of simulation.* Wiley and Sons, New York, 1998.

[4] P. Bratley, B.L. Fox, and L.E. Schrage. *A guide to simulation.* Springer, New York, second edition, 1987.

[5] G. S. Fishman. *Principles of Discrete Event Simulation.* John Wiley and Sons, New York, 1978.

[6] W. D. Kelton, R. P. Sadowski, and D. A. Sadowski. *Simulation with Arena.* McGraw-Hill, New York, second edition, 2002.

[7] A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis.* McGraw-Hill, New York, third edition, 2000.

[8] J. Ch. Strelen. Median confidence intervals. In E.J.H. Kerkhoffs and M. Snorek, editors, *Modelling and Simulation 2001 - Proceedings of the ESM 2001*, pages 771–775. Society for Computer Simulation, 2001.

[9] J. Ch. Strelen. Median confidence intervals - grouping data into batches and comparison with other techniques. In M. Ades and L.M. Deschaine, editors, *Proceedings of the Business and Industry Symposium - ASTC 2002*, pages 169–175. Society for Computer Simulation, San Diego, 2002.

[10] P.D. Welch. The statistical analysis of simulation results. In S.S. Lavenberg, editor, *Computer Performance Modeling Handbook*, pages 267–329. Academic Press, New York, 1983.

MANUFACTURING DATA VALIDATION THROUGH SIMULATION

Guodong Shao
Y. Tina Lee
Charles McLean
Manufacturing Simulation and Modeling Group
National Institute of Standards and Technology (NIST)
100 Bureau Drive, MS 8260
Gaithersburg, Maryland 20899, U.S.A.
E-mail: gshao@cme.nist.gov

KEYWORDS
Computer Aided Manufacturing Engineering (CAME), Data Validation, Numerical Control program, Manufacturing Simulation.

ABSTRACT

In order to ensure that the process plan and numerical control (NC) programs are complete and correct before jobs are released to the shop floor, manufacturing industry needs a validation system that has visualization and analysis capabilities. A prototype manufacturing data validation system has been developed by the Manufacturing Simulation and Modeling Group of the National Institute of Standards and Technology (NIST). The work was carried out as a part of the NIST/NAVY Computer Aided Manufacturing Engineering (CAME) program. The manufacturing data validation system uses simulation and an interactive checklist to validate the manufacturing engineering data package. This paper discusses the manufacturing data validation methodology through a simulation model and presents a prototype of a manufacturing data validation system.

DISCLAIMER

No approval or endorsement of any commercial product by the National Institute of Standards and Technology is intended or implied. Certain commercial software systems are identified in this paper in order to facilitate understanding. Such identification does not imply that these software systems are necessarily the best available for the purpose.

INTRODUCTION

In order to ensure that the process plan and NC program are complete and correct before jobs are released to the shop floor, manufacturing industry needs a validation system that has visualization and analysis capabilities. The manufacturing data validation system uses simulation and an interactive checklist to validate the manufacturing engineering data package.

A manufacturing engineering data package is a set of information necessary for supporting manufacturing operations and resources to produce a part or batch of parts. A data package may include the following types of documents:

- Part specifications and drawings
- Routings and operation sheets
- NC programs
- Tool and fixture lists
- Setup instructions and illustrations
- Material lists
- Quality control data

The degree of accuracy of these manufacturing data is directly related to the quality and cost of the product; therefore, manufacturing engineering data validation is a critical step in the production process. Manufacturing engineering data packages often contain errors that may result in inefficient use of production resources. The erroneous data must be corrected prior to the first production run. Individual data packages and associated production processes need to be checked prior to production startup. For example, an NC program, a part of the manufacturing data, should be validated. An NC program contains a set of instructions; those are the sequences of machining operations on the workpiece. Traditionally, once an NC program is generated, it was validated by checking the NC tool path plots or performing a dry run on a real machine. These traditional validation methods may not be reliable and could be labor intensive, dangerous and costly. Some undetectable errors may remain in the NC program. Those undetected errors may result in poor part quality, premature tool wear out, or tool breakage. A manufacturing data validation process is therefore recommended.

Based on the set of requirements identified for the manufacturing data validation system (Leong and Smith 1995), Delmia's Virtual Numerical Control (VNC), version 4.1 was selected as the development tool for the prototype of the manufacturing data validation system.

IMPLEMENTATION METHODOLOGIES

The primary function of the data validation system is to validate manufacturing process data in a virtual manufacturing environment that emulates a physical manufacturing system. The steps of this validation process include:

- Control and management of the validation process
- Creation of the validation environment

- Validation of data availability
- Validation of data integrity
- Validation of process capability
- Validation of process performance
- Validation of machining parameter
- Detection of collision
- Identification of force and thermal effects
- Evaluation of geometry and tolerances

Data validation methodology

The data validation system ensures that the part will be produced correctly the first time. The data validation can be performed manually or automatically. The elements of validation are described as follows (McLean et al. 1995):

- Data availability: Check whether all required documents exist in the manufacturing data package
- Data integrity: Check whether all documents have the proper structure and information contents
- Resource availability: Check whether all required manufacturing resources exist for the assigned tasks
- Process capability: Check whether the selected resources and processes are capable of achieving the required results without causing any undesirable side effects
- Process performance: Check whether the planned production process achieves quality, time, and cost objectives

NC program validation methodology

The NC program validation verifies the NC program generated by the process planning system. The NC program is the code used to produce the designed part on a specific machine using selected resources. The validation code is implemented as a work element. NC programs are validated in the machine tool simulation model.
The NC program validation ensures:

- Correct media and version
- Cutting operations within proper process envelope
- Rigidity of setup, tools, etc.
- No collisions
- Correct part geometry produced

Before the start of the NC program validation, the following validation procedures need to be performed.

- Operation plan validation
- Machine loading procedure validation
- Workpiece loading procedure validation
- Machine operation procedure validation

The validation procedures are grouped into several functional program units. These program units are sets of subroutines. Wherever possible, standard system, database, or other library routines should be used to implement these functional units.

A PROTOTYPE VALIDATION SYSTEM

A prototype validation system has been developed using the validation methodology. The primary function of the prototype validation system is to simulate machining operations and execute validation procedures as the operation is being executed. The purpose is to assess the feasibility of implementing the validation concept in the simulation environment. The system, shown in Figure 1, is a simulation model for the machining center MAKINO A55 (Delmia 1996). The simulation model incorporates the advantages of program tryout on the actual machine tool, with the ease and speed of computer simulation. Through the emulation of the entire machine tool and Computer Numerical Control (CNC) controller, the prototype model provides the manufacturing data validation system with the following list of capabilities:

- Validate and debug the NC program on a virtual manufacturing environment
- Verify tooling and fixture setups
- Evaluate the post processor and virtual machine models
- Validate and optimize the process and create final manufacturing documentation

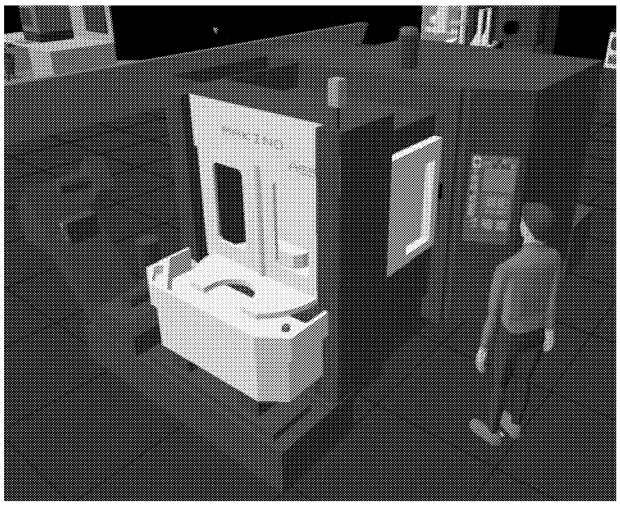


Figure 1: Prototype of the Validation System
- MAKINO A55 Model

In the prototype validation system environment, the following virtual system elements are developed:

- A simulated work cell model of high-speed machining center (Makino A55)
- Models of a tool set for the selected test part
- Workpiece models for test part
- Fixture component models
- A set of validation procedures
- An Oracle database
- Reference data (includes cutting speed table, feed rate table, specific energy table, tool table, spindle power specification given in the machine tool reference manual)

The inputs of the data validation system are:

- NC programs generated by the process planning system
- Process planning data

The outputs of the validation system are:

- Validated NC programs
- Validation reports
- Warning messages
- Error messages
- Finished virtual products

Validation and Simulation System Architecture

The validation and simulation system architecture is shown in Figure 2. Some validation procedures, such as collision control, can be set up in the simulation environment. However, most of the procedures are programmed inside the system control program, a Delmia Mimic program. The procedures are logically related groups of software subroutines.

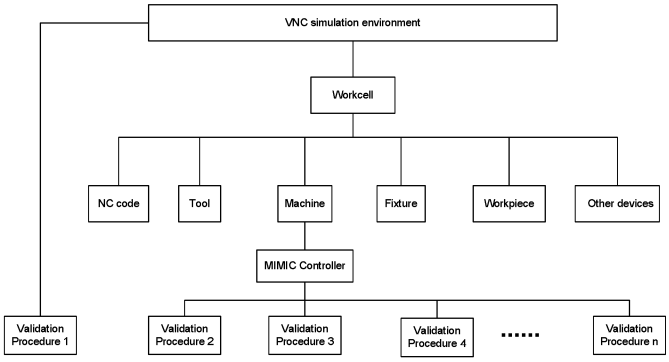


Figure 2: Validation and Simulation Subsystem Architecture

NC Program Validation Procedure

In order to validate the NC program, it needs to be loaded into the virtual machine model, and the simulation needs to be started. The validation program will parse and run the NC statements one by one. The NC statements contain data necessary for the cutting operation such as miscellaneous functions (M codes), rapid motion (G00), or machining operations (G01, G02 and G03), as well as cutting attributes as follows:

- Depth of cut
- Width of cut
- Linear feed rate
- Spindle speed
- Tool identification
- Tool material type
- Tool diameter
- Tool length
- Number of cutting edges

Related information needs to be retrieved from the NC statement and recommended ranges need to be computed, and then checked to ensure that they are within the ranges. If any of the data is out of the recommended range, an error report needs to be generated, and at the same time, a screen message is also displayed.

Required information.

The prototype is focused on milling operations used in the Makino A55 machining center. The checklists are currently limited to high-speed steel and carbide cutting tools, and the part materials are aluminum, titanium, and a few other types. In the process of data validation, some of the required information can be directly obtained. Table 1 lists the method to retrieve the information.

Table 1: Data items and Description of the Required Information

Data item	Description
Tool specifications	Based on the unique tool id listed in the NC program, the cutting tool material, tool diameter (D), tool length (L) and the number of cutting edges can be obtained from the tool database.
Depth of cut	VNC continuously provides the depth of cut of each NC statement.
Width of cut	In a milling operation, the width of cut can be either the difference of two X's or Y's values depending on the machining axis.
Linear feed rate	Linear feed rate (F) can be obtained from the NC statement. It is valid for subsequent NC statements until a new value is listed.
Spindle speed	Spindle speed (S) can be obtained from the NC statement.
Material removal rate	VNC provides function to obtain the material removal rate.
Recommended machining parameters	Recommended machining parameters such as cutting speed, feed rate per cutting edge, or specific energy can be obtained from reference books.

Computation of the machining parameters

For the information that cannot be retrieved directly from the system or NC statements, computations are needed. Also in order to ensure those recommended values are consistent with the unit of measure given in the NC program, unit conversion/computation is often performed. These computations include: converting cutting speed (V) to spindle speed (S), computing linear feed rate (F), and spindle power (N_m) (Serope 1995).

Checklist of the validation for an NC Statement

The NC statement validation will ensure the data and data syntax are correct, complete, and collision free when the machine is in rapid motion. It also ensures the machine

parameters are within specification limits. The NC program's validation checklist is listed as follow:

- M code: The validation ensures that necessary miscellaneous functions, such as coolant on, coolant off, and spindle on, are in the right positions. Table 2 presents the miscellaneous functions checklist items and description.
- Potential collision in rapid motion mode: When the machine is executing G00, the cutting tool is at maximum feed rate. There is no machining operation that occurs during the movements. Therefore, there should not be any contacts between the cutting tool and the machine, workpiece, and fixtures.
- Machining functions: G01, G02, G03, and canned cycles (G81 to G89) are machining cycles. There are several machining parameters associated with these cycles such as spindle speed, linear feed rate, spindle power, maximum depth of cut as listed in Table 3, and maximum step-over depth of the cut. This validation ensures that these machining parameters are within the specification's ranges.

Table 2: Miscellaneous Functions Checklist Items and Description

Item	Description
Spindle starts (M03/M04)	Spindle should be turned on only after a tool has been moved to a desired position.
Coolant on (M08)	Coolant should be turned on only after spindle has been turned on in order not to block operator's view while the spindle is moving to its desired position.
Unlock rotary table (M11)	Table needs to be unlocked before it rotates to avoid damaging the machine, NC statement may command a rotation of the table even when the table is in a locked position.
Lock rotary table (M10)	For accuracy purposes, the rotary table needs to be locked in position after it has been rotated to a desired position.

Table 3 Machining Parameters Checklist Items and Description

Item	Description
Spindle speed	Check if the value is within the recommendation range; spindle speed affects cutting tool's life.
Linear feed rate	Check if the value given in NC program is within calculated range.
Spindle power	Check if the value given in NC program is within the machine specification.
Depth of cut	Check if the depth of cut is between zero and 0.8 of the cutting length interval ($0<W \leq 0.8L$).
Maximum stepover of Cut	Stepover of the cut is the amount of cutter movement for each cut. This validation is to ensure that maximum stepover of the cut does not exceed the recommended values that are between zero and 0.8 of the cutting tool diameter ($0<W\leq 0.8D$).

Validation Report.

At the end of the validation process, a validation report will be generated. The validation report includes:

- Warning messages
- Error messages
- Machining parameters

An error message is generated when data in a given NC program is out of range. A warning message is generated when data is within range but close to the specification limits.

CONCLUSION

This paper has provided a brief overview of the manufacturing data validation methodology through simulation modeling and presented a prototype of a manufacturing data validation system. This research work was developed as a part of the manufacturing engineering tool kit project by staff of the NIST Manufacturing Simulation and Visualization Program. Manufacturing industry needs a validation system that has visualization and analysis capabilities. The manufacturing data validation prototype system presented in this paper uses simulation and an interactive checklist to validate the manufacturing engineering data package and ensure that the process plan and NC program are complete and correct before jobs are released to the shop floor.

ACKNOWLEDGEMENTS

This paper describes the work about the CAME toolkit research effort by the Manufacturing Simulation and Modeling Group at NIST. Authors especially thank Swee Leong, Vuong Nguyen and C.C. Chen for their contribution of the research work and ideas in this paper.

REFERENCES

Delmia. 1995. *Virtual NC Reference Manual*. Delmia, Auburn Hills, MI.

Leong, S. and M. Smith. 1996. "Computer-Aided Manufacturing Engineering Forum Second Technical Meeting Proceedings".

NISTIR 5846, National Institute of Standards and Technology, Gaithersburg, MD

Makino. 1998. *A55 Plus, High Efficiency Horizontal Machining Center*: *Operation Guide*. Japan.

McLean, C. R.; S. Leong; and C. Chen. 1995. "Issues of manufacturing Engineering Data Validation in Concurrent Engineering Environment". In *Proceedings of IFIP WG5.7 Working Conference on Managing Concurrent Manufacturing to Improve Industrial Performance*. Seattle, WA.

Pusztai, J. and M. Sava. 1983. *Computer Numerical Control*. Reston Publishing Company, Reston, VA.

Serope, K. 1995. *Manufacturing Engineering and Technology.* Third Edition. Addison Wesley Publishing Company, Reading, MA.

BIOGRAPHY

GUODONG SHAO is a computer scientist in the Manufacturing Simulation and Modeling Group at NIST. He has participated in research on Flexible Manufacturing System (FMS), Computer Intergrated Manufacturing Syatem (CIMS), Computer Aided Manufacturing (CAM), Shop Floor Control, Product Data Management (PDM), process planning, manufacturing simulation and manufacturing system integration for many years. He has hands on experience on most of the simulation packages currently in the market. He holds a Master's Degree from University of Maryland at College Park. He is a Ph.D. student in Graphics Laboratory at George Mason University. His e-mail address is gshao@cme.nist.gov.

Y. TINA LEE is a computer scientist in the Manufacturing Simulation and Modeling Group at NIST. She joined NIST in 1986. Most recently, she has been working on the design and development of interface information models to support the Software Engineering Institute (SEI) Technology Insertion Demonstration and Evaluation (TIDE) project. Previously she worked at the Contel Federal Systems and at the Sperry Corporation. She received her BS in Mathematics from Providence College and MS in Applied Science from College of William and Mary. Her e-mail address is leet@cme.nist.gov.

CHARLES MCLEAN is a computer scientist and Program Manager of the Manufacturing Simulation and Visualization Program at NIST. He also leads the Manufacturing Simulation and Modeling Group. He has managed research programs in manufacturing simulation, engineering tool integration, product data standards, and manufacturing automation at NIST since 1982. He has authored more than 50 technical papers on topics in these areas. He is on the Executive Board of the Winter Simulation Conference and the Editorial Board of the International Journal of Production, Planning, and Control. He is formerly the Vice Chairman of the International Federation on Information Processing (IFIP) Working Group on Production Management Systems (WG 5.7). He is also the NIST representative to the Department of Defense's Advanced Manufacturing Enterprise Subpanel. He holds a MS in Information Engineering from University of Illinois at Chicago and a BA from Cornell University. His e-mail address is mclean@cme.nist.gov.

TOWARDS PERFORMANCE EVALUATION OF MOBILE SYSTEMS IN UML

Simonetta Balsamo
Moreno Marzolla

Dipartimento di Informatica
Università Ca' Foscari di Venezia
via Torino 155, 30172 Mestre (VE), Italy
e-mail: {balsamo|marzolla}@dsi.unive.it

Keywords

Process Oriented Simulation, Software Architectures, Unified Modeling Language (UML), Performance Evaluation, Mobile Systems.

Abstract

The current generation of network-centric applications exhibits an increasingly higher degree of mobility. Wireless networks allow devices to move from one location to another without loosing connectivity. Also, new software technologies allow code fragments or entire running applications to migrate from one host to another. Performance modeling of such complex systems is a difficult task, which should be carried out during the early design stages of system development. However, the integration of performance modeling analysis with software system specification for mobile systems is still an open problem, since there is no unique widely accepted notation for describing mobile systems. Moreover performance modeling is usually developed separately from high-level system description. This is not only time consuming, but the separation of performance model and system specification makes more difficult the feedback process of reporting the performance analysis results at the system design level, and modifying system model to analyze design alternatives. In this paper we address the problem of integrating system performance modeling and analysis with a specification of mobile software system based on UML. In particular we introduce a unified UML notation for high-level description and performance modeling of mobile systems. The notation allows inclusion of quantitative information, which are used to build a process-oriented simulation model of the system. The simulation model is executed, and the results are reported back in the UML notation. We describe a prototype tool for translating annotated UML models into simulation programs and we present a simple case study.

1 INTRODUCTION

The current generation of network-centric applications exhibits an increasingly higher degree of mobility. From one side, wireless networks allow devices to move from one location to another without loosing connectivity. From the other side, new software technologies allow code fragments or entire running applications to migrate from one host to another. In this direction, design approaches based on location awareness and code mobility have been proposed where the application components can move to different locations during their execution. This should improve system quality, allowing a higher degree of flexibility and increasing its performance. Indeed, from the performance viewpoint, moving application components in a distributed environment could lead to transforming remote interactions into local ones. We consider software mobile system at a high level of abstraction and we refer to Software Architecture (SA) to describe system structure and behavior (Bass et al., 1998; Shaw and Garlan, 1996). In particular, one can define mobile SA with various mobility styles, whose definition depends on whether they require copies creation of components at new locations, or local change of components preserve their identity (mobile agent). In the former case we can further distinguish systems where the copy is created at the location of the component that starts the interaction (code on demand), or systems where it is a created at the location of the component that accept the interaction (remote evaluation). Many formalisms to represent mobile software systems and reasoning about mobility have been proposed (De Nicola et al., 1998; Milner, 1999; Nottegar et al., 2001; Picco et al., 2001). However, most of them cannot be considered architectural description languages (ADL), since they do not explicitly model components and interactions as first class entity.

Several models of SA have been proposed based on formal ADL with precise semantics and syntax (Medvidovic and Taylor, 2000). On the other side, due to the difficulties in integrating formal ADL in the design practice, other approaches consider semi-formal widely used modeling languages such as the Unified Modeling Language (UML) (Medvidovic et al., 2002; Object Management Group, 2001; Rumbaugh et al., 1999).

In this paper we consider mobile software systems at the SA level, and an UML-based system specification. Performance modeling of mobile software systems is a difficult task, which should be carried out from the early design stages of system development. The integration of quantitative performance analysis with software system specification has been recognized to be a critical issue in system design. Per-

formance is one of the most influential factors that drive system design choices. Several approaches have been proposed to integrate performance evaluation tools in the early stages of the software development life cycle (Balsamo et al., 2002; Smith and Williams, 2002). However, few of them consider mobile systems (Cortellessa and Grassi, 2002) and there is still a lack of tools for performance analysis that support the system designer in the selection of SA of mobile software systems to meet given performance requirements.

We focus on the integration of performance modeling analysis with software system specification for mobile systems. To this aim we observe that there is not a unique widely accepted notation for describing mobile systems. Moreover performance modeling is often developed separately from high-level system description. This is not only time consuming, but the separation of performance model and system specification makes more difficult the feedback process of reporting the performance analysis results at the system design level, and modifying system model to analyze design alternatives.

In this paper we address the problem of integrating system performance modeling and analysis with a specification of mobile software system based on UML. In particular we propose an integrated approach to modeling and performance evaluation of mobile systems at the architectural level based on simulation. We consider both physical mobility (devices which physically change their locations) and code mobility (code fragments which migrate from one execution host to another).

The main advantage of using a simulation approach is that it allows a high degree of model flexibility, a direct representation of system components and makes it easy to report performance results back at the SA design level. Indeed, various approaches based on simulation models of software systems have been recently proposed to evaluate SA performance (Arief and Speirs, 1999, 2000; De Miguel et al., 2000). Simulation greatly simplifies the derivation of the performance model from the software system specification, with respect to other approaches based on analytical models. However, code mobility has not been considered in these proposals.

We introduce a unified UML notation for high-level description and performance modeling of mobile systems. The software system description based on the standard UML notation is integrated with performance-oriented parameters, which are described with the UML extension mechanisms (namely stereotypes and tagged values). The notation allows users to include quantitative information, which are used to build a process-oriented simulation model of the system. We use a subset of the annotations proposed in the UML Performance Profile (Object Management Group, 2002a). Then, the annotated UML diagrams are converted into a process-oriented simulation model. The simulation model is executed to evaluate the performance of the mobile system, and simulation results are reported back the original UML diagrams as tagged values. We describe a prototype tool for translating annotated UML models into simulation programs and we present a case study.

This paper is organized as follows. Section 2 introduces the proposed approach to mobility modeling with UML Use Case, Activity and Deployment diagrams. Section 3 describes the annotations used for performance modeling of UML specifications. Section 4 presents a simple case study of the proposed methodology. Conclusions and open research are discussed in Section 5.

2 UML MOBILITY MODELING

We consider mobile software systems at the SA level, and an UML-based system specification. UML is a standard graphical notation for modeling object-oriented software, and it has been widely applied in the software development process (Object Management Group, 2001; Rumbaugh et al., 1999). The graphical notation provides several types of diagrams that represent different system views of the system. UML provides extension mechanisms that include stereotypes and tagged values, to integrate the need of specific domains, We observe that currently there is no standard way for expressing mobility in UML, although different proposals exist in the literature.

2.1 Previous work

Baumeister et al. propose in (Baumeister et al., 2003) an extension of UML Class and Activity diagrams to represent mobility. They define new stereotypes for identifying mobile objects and locations. Stereotypes are also defined for moving and cloning activities. Mobile systems are then represented by Activity diagrams using either a "responsibility centered notation", which focuses on who is performing actions, and a "location centered notation" which focuses on where actions are being executed and how activities change their location. While this approach has the advantage of requiring only minor extensions to UML, a possible shortcoming is that it represents in the same Activity diagram both the mobility model (how objects change their location) and the computation model (what kind of computations the objects perform). For large models this could render the diagrams difficult to understand.

Some UML notation mechanisms can be used to represent mobile SA, as discussed in (Rumbaugh et al., 1999). They are based on the tagged value `location` to express a component location, and the stereotypes `copy` and `become` to express the location change of a component. They can be used in Collaboration diagrams to model location changes of mobile components. Grassi and Mirandola (Grassi and Mirandola, 2001) suggest an extension to UML to represent mobility using Collaboration diagrams. Collaboration diagrams contain a `location` tagged value representing the physical location of each component. They define the `moveTo` stereotype, which can be applied to messages in the Collaboration diagram. When the `moveTo` stereotype is present, it indicates that the source component moves to the location of the destination component before interacting with it. Sequence diagrams are used to describe the interactions between components, regardless of their mobility pattern.

Kosiuczenko (Kosiuczenko, 2002) proposes a graphical

notation for modeling mobile objects based on UML Sequence diagrams. Mobile objects are modeled using an extended version of lifelines. Each lifeline is represented as a box that can contain other objects (lifelines). Stereotyped messages are used to represent various actions such as creating or destroying an object, or entering and leaving an object. This approach has the drawback of requiring a change in the standard notation of UML Sequence diagrams, that is, lifelines should be represented as boxes, with possibly other Sequence diagrams inside. Existing graphical UML editors and processors need to be modified in order to support the new notation.

2.2 The approach

We model a mobile system as a collection of devices, which can be either processors or communication links. A computation on the system is modeled as a set of activities carried out on the devices. A configuration of the system is a specific allocation of activities on processors. So, while a mobile entity travels through the system, it activates a sequence of configurations, each representing a specific system state. Mobile entities may be both physical devices traveling in the real space, or software components which migrate from one processor to another. Once a configuration is activated, the mobile entity starts an interaction with the system. This typically includes requesting service to the devices (processors) or performing communications, which we also model as requesting service to network devices. We assume that while a mobile entity is interacting with the system, it cannot move, i.e., the system configuration cannot change. Further movements are possible when the interaction is completed.

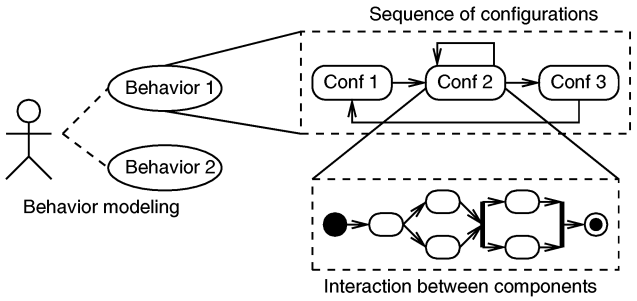


Figure 1: Overview of the mobility modeling methodology

The proposed approach to mobile system modeling involves three main steps, which are depicted in Fig. 1

1. Enumerate the various mobility strategies of the mobile entities.
2. Model the sequence of configurations which are triggered in each mobility strategy.
3. Model the interactions performed by the mobile entities in each configuration.

The first step deals with identifying the mobility pattern of the entities. For example, users of the system may exhibit a bigger or smaller probability to change their location, hence they may exhibit different degrees of mobility. Also, users may move through the system with different preferential patterns. We define a *mobility behavior* as the sequence of different configurations which are executed while the user moves. Such mobility behaviors need to be identified, and further described in the next steps. Mobility behaviors are represented by UML Use Case diagrams. Also, the set of resources (processors) which are present in the system are described using Deployment diagrams.

The next step involves the description of the sequence of configurations which are triggered while the mobile entities travel through the system. Such description can be easily expressed as a state transition diagram. We use Activity diagrams for this purpose. Each activity represents a particular configuration of the system. Transitions describe the order in which configurations are triggered as the user moves. We will refer to these diagrams as "high level" Activity diagrams.

The last step involves detailing what happens while the system is in each configuration. This means specifying what are the interactions between the components while each configuration is active. This is done again using Activity diagrams. Each Action state is associated to the Deployment node instance on which the activity takes place. Each node of the Activity diagrams defined in the previous step is expanded as an interaction. This can be readily expressed using standard UML notation as Activity diagrams have a hierarchical structure, that is, each action state may be exploded into another diagram. We will refer to these diagrams as "low level" Activity diagrams.

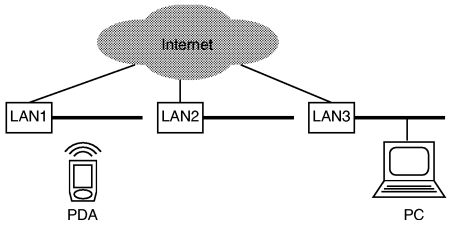
In order to illustrate the proposed approach we introduce an application example of software mobile system. Let us consider the example of software system illustrated in Fig. 2. There is a mobile user that is connected to a PC using a PDA with a wireless network card. The user is viewing a video stream, which is generated by a video server residing on the PC.

Three different Local Area Networks (LAN1, LAN2 and LAN3) are connected through the Internet. Each LAN allows wireless connections as well as wired ones. The PC is connected to LAN3 and does not move, while the user with the PDA travels through the different LAN. In the configuration C_1 of Fig. 2(a) the communication between the PDA and the PC travels through the path LAN1–Internet–LAN3. In the configuration C_2 of Fig. 2(b) the communication is routed through the path LAN2–Internet–LAN3, and in the configuration C_3 of Fig. 2(c) the communication between the PC and the PDA is routed through LAN3 only.

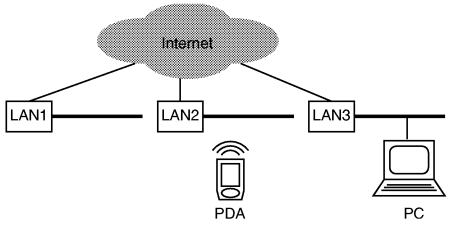
2.3 Modeling the choice of Mobility

As the very first step, it is necessary to provide the physical structure of the system. This can be done by using UML Deployment diagrams, which describe the processing resources available on the system. Such resources include both CPUs and also communication links. The Deployment diagram describing the system in the example is illustrated in Fig. 3.

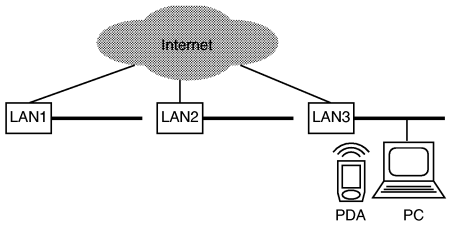
In the example above, we suppose that the mobile user can behave in two different ways. In behavior B_1 he joins the



(a) Configuration C_1



(b) Configuration C_2



(c) Configuration C_3

Figure 2: A mobile user travels through three different LAN

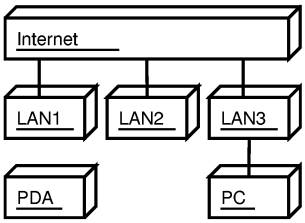


Figure 3: Deployment diagram for the example

system in LAN1, then travel to LAN2 and finally to LAN3 and leaves the system. In behavior B_2 the user joins the system in LAN2, travel to LAN1 and leaves the system. We model these behaviors with the Use Case diagram in Fig. 4.

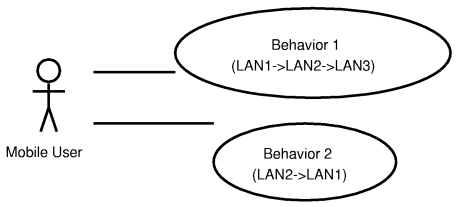
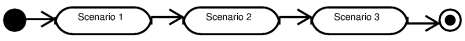


Figure 4: UML representation of different mobility possibilities

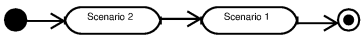
The diagram shows an Actor (mobile entity) that can perform one of the associated Use Cases, each representing one possible mobility behavior. An Actor represents each class of mobile entity. The Use Cases associated with that Actor represent the different ways in which entities of the associated mobile entity class may interact with the system. Note that this is perfectly consistent with the UML semantics of Use Case diagrams (Object Management Group, 2001), as they are used to specify the behavior of an entity without specifying its internal structure.

2.4 Modeling Mobility Behaviors

The next step is to describe the order in which configurations are activated in each behavior. To do that, we associate an Activity diagram to each Use Case; each activity of the Activity diagram represents a configuration of the system. If the mobile user triggers the configuration C_j immediately after the configuration C_i, then in the Activity diagram there will be a transition between the activity representing C_i and the one representing C_j. Considering our example, the two behaviors B_1 and B_2 are represented as the Activity diagrams of Fig. 5.



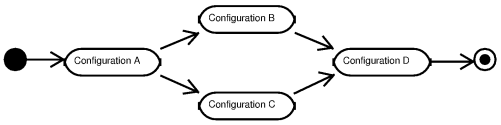
(a) Activity diagram associated to B_1



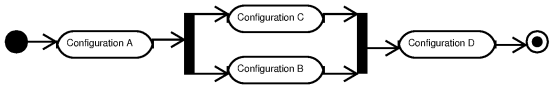
(b) Activity diagram associated to B_2

Figure 5: Activity diagrams associated to the mobility behaviors

Note that in this way it is possible to represent nondeterminism that is a behavior can have multiple successors. Fig. 6(a) illustrates an example where the mobile entity starts by activating configuration A. Then it may proceed by activating one of configuration B and C. After that, configuration D is activated. It is important to observe that the Activity diagrams associated to behaviors do not need to be acyclic. Thus, it is also possible to model situations in which the user triggers the same sequence of configurations for a number of times. Moreover, using fork and join nodes of Activity diagrams it is possible to represent the concurrent execution of different configurations. This can be used to model situations in which the mobile entity generates copies of itself, each one traveling independently through the system. Fig. 6(b) shows an example where a mobile entity starts by entering configuration A. Next, it splits in two copies, one executing configuration B and the other executing configuration C in parallel. This means that the two copies of the mobile entity can move to different locations and perform different interactions with the system. After that, the two copies synchronize and collapse into one instance, which proceeds by executing configuration D.



(a) Nondeterministic behavior



(b) Concurrent agent execution

Figure 6: Modeling nondeterministic mobility behavior and mobile concurrent execution of multiple agent instances

2.5 Modeling Component Interactions

The final step is describing the activities carried out in each configuration. To do that, we use the hierarchical structure of the Activity diagrams to associate the interactions to each action state identified in the previous step. Namely, we expand each action step representing a configuration into the Activity graph describing the sequence of actions which are taken during the interaction between the mobile entity and the system. It is necessary to specify where the actions are executed. To do so it is possible to use "swimlanes", which are a means for specifying responsibility for actions. The name of the swimlanes denotes the Deployment node instance on which the actions execute. As some graphical UML editors do not support swimlanes, it is possible to tag each action with the `PAhost` tagged value, whose value is the name of the node instance of the Deployment diagram corresponding to the host where the action is executed. Fig. 7 shows the interactions performed while the considered system is in configuration C_1 and C_3 using the swimlane-based notation.

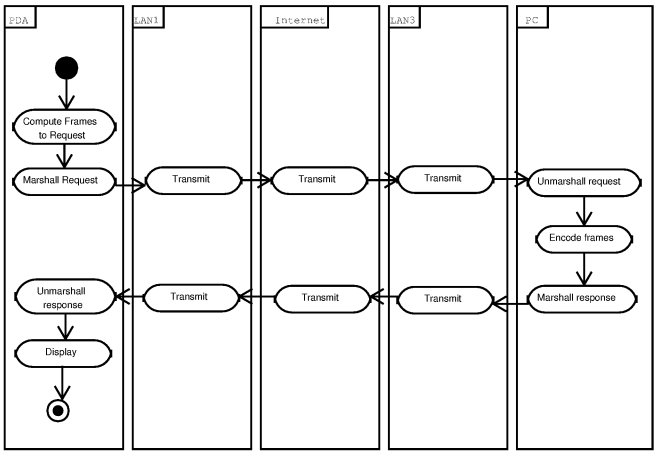
The interaction between the PDA and the PC is very simple. Basically, first the PDA computes which frames it needs. Then a suitable request is encoded and sent through the communication networks to the PC. The request is unmarshaled, and the requested frames are encoded and packed into a reply message. This message is sent back through the network to the PDA, which finally displays the frames. Note that in Fig. 7 we omitted the description of the interaction in configuration C_2, as this is basically the same as in C_1, with the only difference that LAN2 is used instead of LAN1.

2.6 Summary of the Methodology

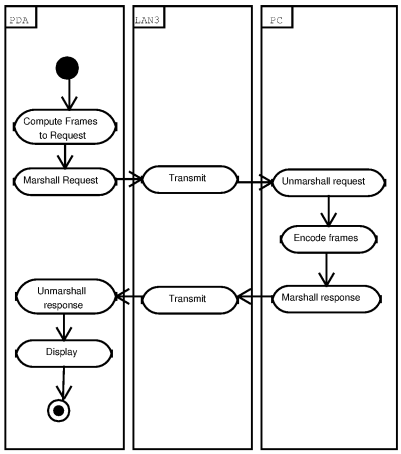
The proposed mobility modeling methodology can be summarized in the following steps:

1. Identify the processing resources (processors or networks) available in the system. Each resource is represented by a node instance in the UML Deployment diagram.

2. Identify the classes of users of the system. Users represent workloads applied to the system. Each class of users is represented as an Actor in the Use Case diagram. Actor may represent either a fixed population of users (closed workload) or an unlimited stream of users (open workload).



(a) Interaction in configuration C_1



(b) Interaction in configuration C_3

Figure 7: UML description of the interactions

3. Identify the mobility behaviors. For each class of users it is necessary to identify the different pattern of mobility they may exhibit. Each of such mobility patterns is represented by a Use Case associated to the Actor representing the class of users.

4. Provide a high level description of the mobility behaviors. An Activity diagram is association to each of the Use Cases identified in the previous step. Such Activity diagram represents the sequence of configuration changes which happens in the system while the mobile user moves.

5. Describe the interactions occurring in each system configuration. Each Action states defined in the previous step are expanded into a low-level Activity diagram describing the interactions between system entities. Each Action of the low-level diagram represents a service requested to a specific processing resource. Code mobility

is represented by associated Activities in the low-level diagram to different hosts. Physical mobility is represented by a possibly different interaction pattern associated to nodes of the high-level Activity diagram.

Quantitative informations required by the simulator can be associated to UML elements as described in the following.

3 UML PERFORMANCE MODELING

We shall now summarize the proposed approach to integrate performance modeling with UML specifications for software mobile SA. The approach derives the performance model directly from annotated UML specifications extended to include mobility as described in the previous Section. We consider the performance model and the annotations based on those proposed in the UML Performance Profile (Object Management Group, 2002a). The profile has been defined using standard UML extension mechanisms, and provides the modeler with a set of packages. Each package defines the mapping between specific domain models (schedulability, time and performance characteristics) to UML stereotypes and tagged values definitions, which represent the UML viewpoint.

We consider SA described in terms of Use Case, Activity and Deployment diagrams. We derive a process oriented simulation model of the software system that is then executed to derive performance indices. Finally, simulation results are reported back into the UML diagrams as tagged values, so they are readily available at the software designer, integrated in the UML system specification.

We now briefly sketch how the performance model is derived; a more detailed description of the derivation of the simulation model from UML diagrams for systems without mobility can be found in (Balsamo and Marzolla, 2003a,b). In the following we show how the methodology can be directly applied to modeling code and physical mobility of the application described by the SA.

Fig. 8 illustrates the structure of the performance simulation model derived from the UML diagrams. The basic object of the simulation model is a `PerformanceContext`. This object contains the other elements of the model, namely Workloads, Scenarios and Resources. Hence there are three types of simulation processes derived from the UML diagrams that are Workload, Scenarios and Resource processes.

Workloads can be open or closed, depending on whether the number of users accessing the system is unbounded or fixed. Each Workload actually drives one or more Scenarios. Each time a new user belonging to a Workload requests service to the system, one of the Scenarios associated with that workload is selected. Selection is done randomly, according to the probability associated to each scenario.

A Scenario is a set of abstract scenario steps, represented by the `AbsStep` class. Abstract scenario steps can either be composite steps (described by the `PScenario` class), or atomic steps of different kinds. Scenarios are collections of steps; exactly one of these steps is marked as the root step (starting step) of the scenario. Atomic steps can be of type `PStep_fork` for nodes representing the creation of multiple execution threads, `PStep_join` for nodes representing synchronization points between different threads, and `PStep` for normal atomic steps.

Each Scenario step executes on a single processor, to which it requires service. Processors are modeled by objects of type `AbsPRhost`. A processor is characterized, among other things, by a scheduling policy which can be one of "FIFO" (first come first served), "LIFO" (last come first served) or "PS" (processor sharing). The simulation model defines some classes derived from `AbsPRhost` which implement a specific scheduling policy.

Each UML model element (Actor, Use Case, Activity state, Node instance) can be tagged with additional quantitative information, which is necessary to derive the parameters of the simulation model. Examples of such parameters are the interarrival time of users, the service demand of action steps, the speed factor of each processor.

Concerning mobility modeling, each Use Case are tagged with the probability of its occurrence, that is, the probability that the associated Actor (motile entity) will execute that Use Case (mobility behavior) upon arriving to the system. Action states of the high-level Activity diagram associated to each Use Case are annotated with the probability of occurrence, the number of times they are repeated and the delay between repetitions. As each action state represents a configuration, the annotations allow the specify (nondeterministically) the pattern of mobility and how long the system remains in each configuration. When the simulator "executes" a configuration, it basically executes all the activities of the low-level Activity diagram embedded in the configurations.

A mobile code fragment moving from host H_1 to host H_2 is represented as follows. Let C_1 be the configuration where the code executes in H_1 and C_2 the configuration where the code executed in H_2. The low level Activity diagram describing C_1 and C_2 will contain an action state (or a whole subdiagram) corresponding to the computation. Such action state or subdiagram will be tagged with the `PAhost` tagged value, which describes the location where the activity is executed. In C_1 we set `PAhost`=H_1, and in C_2 we set `PAhost`=H_2.

Physical mobility is modeled in a similar way. If a mobile device travels through the system, as in our example, probably it will interact with different other nodes for communicating. Depending on the situation, it may even choose a different communication pattern, and hence a different interaction style with other entities. Such interaction styles will be represented by (possibly different different) structures of the low level Activity diagrams.

The derivation of the performance model from the UML specification works as follows. Each actor of the Use Case diagrams is translated into an `OpenWorkload` or `ClosedWorkload` simulation process, depending on its associated stereotype. Note that to represent mobility, each Actor represents a class of mobile entity.

Then, each Use Case associated with the Actor is examined and translated into a `PScenario` simulation process.

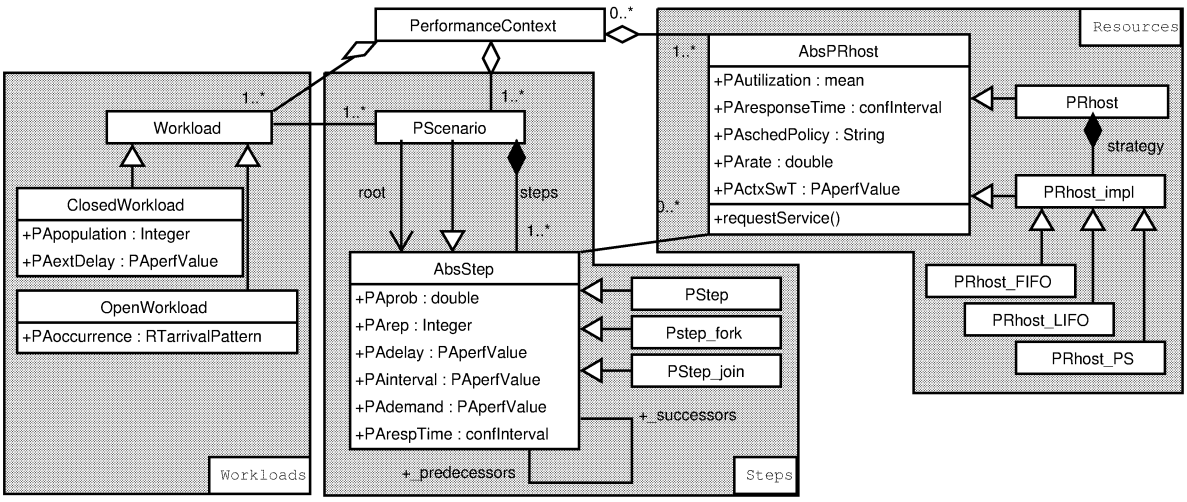


Figure 8: Structure of the simulation performance model

To define the content (sequence of steps) of the scenario, the high level Activity Diagram associated with the Use Case is examined. The structure of such diagram is translated into a network of `PScenario` simulation processes. The internal behavior of such processes is modeled by the low level Activity diagram contained in the high level action stated.

All the activities are translated into the appropriate kind of step (that is, `PStep`, `PStep_fork` or `PStep_join`) depending on the type of the UML element. The transformation is applied recursively if an activity is indeed composite of a sub-activity diagram. Each object derived from the `AbsStep` class is associated with an object representing the physical resource on which it executes. The mapping between actions and the resources where they execute can be derived by examining the `PRhost` tagged value, or the name of the swimlane containing the activity.

Finally, each node instance in the Deployment Diagrams is translated into a simulation process of type `AbsPRhost`.

We developed UML-Ψ (UML Performance SImulator), a prototype performance evaluation tool, which processes an XMI (Object Management Group, 2002b) description of UML Use Case and Activity diagrams. The UML SA has to be annotated using a simplified subset of the UML Performance Profile.

The simulation model is process oriented and its objects are derived by the analysis of the UML diagrams annotated with performance specification of the software system components, and with the extension to describe mobility, as describer in the previous Section. The simulation model is implemented as a discrete-event simulation program written in C++, whose execution provides results for a set of performance indices. We evaluate through simulation the mean response time associated with the execution of each scenario (Use Case) and each scenario step (Activity). Simulation results, i.e., the performance measures of the software components are inserted back into the original UML SA as tagged values to provide feedback to the system designer.

4 CASE STUDY

The case study illustrated in the previous sections has been simulated using the parameters reported on Table 1. A single user interacts with the system (closed workload), and the probability p that the user triggers the behavior B_1 (see Fig. 5) has been set to $p = 0.3$, while the probability that the user triggers the behavior B_2 has been set to $1 - p$.

Parameter	**Value**
(Un)Marshalling Requests	Const. $0.1s$
(Un)Marshalling Responses	Expon. mean=$5.0s$
Request Transmission Times	Exp. mean=$1.0s$
Response Transmission Times	Exp. mean=$10.0s$
Request Computation	Exp. mean=$0.1s$
Frame Encoding Time	Exp. mean=$20.0s$
Display time on the PDA	Exp. mean=$20.0s$
PDA Speedup factor	0.2
PC Speedup factor	10.0
Processors Sched. Policies	FIFO

Table 1: Simulation Parameters. “Exp.” means exponentially distributed with the given mean. “Const.” means constant.

The simulation results are shown in Table 2. They are the steady-state mean values computed at 90% confidence level; for simplicity only the central value of the confidence interval is shown. Such results are automatically inserted into the UML diagram as tagged values associated to the relevant UML elements, as described in (Balsamo and Marzolla, 2003a,b). In this way it is very easy to get immediate feedback on the model performances.

5 CONCLUSIONS

In this paper we introduced an integrated methodology for UML-based modeling and performance evaluation of mobile systems. Only minimal extensions to the semantics of UML are requested in order to apply the methodology. The performance modeling is based on a subset of the UML Perfor-

Internet utilization	0.039
LAN1 utilization	0.010
LAN2 utilization	0.028
LAN3 utilization	0.072
PDA utilization	0.832
PC utilization	0.015
Behavior B_1 response time	445s
Behavior B_2 response time	312s

Table 2: Simulation Results.

mance Profile. We depicted the structure of the simulation performance model which is automatically derived from the annotated UML diagrams. Using a simple case study, we showed the approach can be applied.

Our current research involves the development of a more compact ways to describe the UML diagrams used in the proposed approach. In fact, we note that many of the low level activity diagrams have the same structure, with only minimal differences mostly related to the location of the activities. Describing such diagrams in some "parametric" way would greatly alleviate the work of the modeler.

ACKNOWLEDGMENTS This work has been partially supported by MURST Research Project Sahara and by MIUR Research Project FIRB "Performance Evaluation of Complex Systems: Techniques, Methodologies and Tools".

REFERENCES

L. B. Arief and N. A. Speirs. Automatic generation of distributed system simulations from UML. In *Proc. of ESM '99, 13th European Simulation Multiconference*, pages 85–91, Warsaw, Poland, June 1999.

L. B. Arief and N. A. Speirs. A UML tool for an automatic generation of simulation programs. In Proceedings of WOSP 2000 Proceedings of WOSP 2000, pages 71–76.

S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni. Software performance: state of the art and perspectives. Tech Rep. TR SAH/04, MIUR Sahara Project, Dec. 2002.

S. Balsamo and M. Marzolla. A simulation-based approach to software performance modeling. Tech. Rep. TR SAH/44, MIUR Sahara Project, Mar. 2003a.

S. Balsamo and M. Marzolla. Simulation modeling of UML software architectures. In D. Al-Dabass, editor, *Proc. of the European Simulation Multiconference*, pages 562–567, Nottingham, June 2003b.

L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison-Wesley, 1998.

H. Baumeister, N. Koch, P. Kosiuczenko, and M. Wirsing. Extending activity diagrams to model mobile systems. In M. Aksit, M. Mezini, and R. Unland, editors, *NetObjectDays*, volume 2591 of *Lecture Notes in Computer Science*. Springer, 2003. ISBN 3-540-00737-7.

V. Cortellessa and V. Grassi. A performance based methodology for early evaluate the effectiveness of mobile software architecture. *J. of Logic and Algebraic Programming*, 51: 77–100, Apr. 2002.

M. De Miguel, T. Lambolais, M. Hannouz, S. Betgé-Brezetz, and S. Piekarec. UML extensions for the specifications and evaluation of latency constraints in architectural models. In Proceedings of WOSP 2000 Proceedings of WOSP 2000, pages 83–88.

R. De Nicola, G. Ferrari, R. Pugliese, and B. Venneri. KLAIM: a kernel language for agents interaction and mobility. *IEEE Trans. on Soft. Eng.*, 24(5):315–330, 1998.

V. Grassi and R. Mirandola. UML modelling and performance analysis of mobile software architectures. In M. Gogolla and C. Kobryn, editors, *UML*, volume 2185 of *Lecture Notes in Computer Science*. Springer, 2001. ISBN 3-540-42667-1.

P. Kosiuczenko. Sequence diagrams for mobility. In J. Krogstie, editor, *Proc. of MobIMod workshop*, Tampere, Finland, Oct. 2002. Springer.

N. Medvidovic, D. S. Rosenblum, and D. F. Redmiles. Modeling software architectures in the Unified Modeling Language. *ACM Trans. on Soft. Eng.*, 11(1):2–57, 2002.

N. Medvidovic and R. N. Taylor. A classification and comparison framework for software architecture description languages. *IEEE Trans. on Soft. Eng.*, 26(1):70–93, 2000.

R. Milner. *Communicating and Mobile Systems: The π-Calculus*. Cambridge University Press, 1999.

C. Nottegar, C. Priami, and P. Degano. Performance evaluation of mobile processes via abstract machines. *IEEE Trans. on Soft. Eng.*, 27(10):338–395, 2001.

Object Management Group. Unified modeling language (UML), version 1.4, Sept. 2001.

Object Management Group. UML profile for schedulability, performance and time specification. Final Adopted Specification ptc/02-03-02, OMG, March 2002a.

Object Management Group. XML Metadata Interchange (XMI) specification, version 1.2, Jan. 2002b.

G. P. Picco, G.-C. Roman, and P. J. McGann. Reasoning about code mobility in mobile UNITY. *ACM Trans. On Soft. Eng. and Methodology*, 10(3):338–395, 2001.

Proceedings of WOSP 2000. Ottawa, Canada, Sept. 2000. ACM Press.

J. Rumbaugh, I. Jacobson, and G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 1999.

M. Shaw and D. Garlan. *Software Architecture: perspectives on an emerging disciline*. Prentice-Hall, 1996.

C. U. Smith and L. Williams. *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*. Addison–Wesley, 2002.

SIMULATION OF THE BEHAVIOR OF A NEW MODEL OF REVERBERATING CHAMBER FOR THE EVALUATION OF ELECTROMAGNETIC COMPATIBILITY IN THE TIME DOMAIN

Nicola Pasquino
Department of Electrical Engineering
University of Naples "Federico II"
Via Claudio, 21 – 80125 Naples, Italy
E-mail: npasquin@unina.it

KEYWORDS
Electromagnetic Compatibility, Reverberating Chambers, Chaotic Behavior;

ABSTRACT

Simulations of the behavior of a new model of reverberating chamber to be used in electromagnetic compatibility testing of electronic equipment is presented. The environment is studied resorting to the principles of chaos theory, and results of simulations are provided that show the variations of some quantities of interest – such as isotropy and uniformity of incidence angles – when quantities like dimensions and position of the device under test are varied. Results show that use of the proposed environment is possible as long as some limits are applied to the range of variation of the investigated variables.

INTRODUCTION

The exponential growth of systems that use wireless transmission of energy at either radio or microwave frequency requires that any electronic device be characterized in terms of immunity to external electromagnetic (EM) fields and emission of EM waves into the external environment. The ultimate aim of the science known as Electromagnetic Compatibility (henceforth EMC) is to assess the capability of an instrument or electronic apparatus 1) to operate within design specifications even when it is under the influence of an EM disturbances (immunity characteristics), and 2) to keep the unintentional EM emission at a level that does not corrupt operational characteristics of other devices that share the same environment (emission characteristics), thus comprising both an active and passive role of the object under test.
Tests are performed in a variety of environments like semi-anechoic rooms, TEM (transverse electromagnetic) and G-TEM (Gigahertz TEM) cells, open area test sites (OATS) and shielded rooms. They differ in the kind of tests for which they can be used (i.e., conducted or radiated, immunity or emission tests), the range of frequencies and signal amplitudes that can be generated inside them, the stability over time of EM characteristics and the measurement uncertainty they present.
A reverberating chamber (Corona et al. 1976; Corona et al. 1979; Corona 1980; Corona et al. 1980) is an additional environment which has the interesting property of allowing an isotropic field to be generated in each point inside a volume almost as large as the chamber itself. That means that the object under test is radiated from every direction, which saves time compared to a semi-anechoic room which may be used for a 61000-4-3 immunity test – in which preliminary test should be carried out that lead to an indication of the most sensitive region of the equipment under test – in that it will avoid the preliminary research of the most sensitive configuration to external fields. An additional saving would be obtained if the EUT could be test against the entire range of test frequencies or at least a large subset of it at the same time, maybe through judicious use of an appropriately shaped pulse containing the frequency of interest. This has been the main motivation for the research undertaken and presented in this paper.
We will proceed according to the following outline: in the next section the basic principles of reverberating enclosures will be presented; then, the proposed model of a new reverberating enclosure will be introduced and some preliminary result about its behavior in terms of a cinematic equivalent given. The full electromagnetic model will then be presented and finally results of simulations will be given, and conclusions and suggestions for some future work drawn from them.

REVERBERATING CHAMBERS

A reverberating chamber is a shielded enclosure in which *modal stirrers* or *tuners* – depending on whether their motion is continuous with time or discrete – rotate resulting in a change of the boundary conditions of the electromagnetic problem. The change causes a variation in the modal solutions within the volume of the enclosure. If the geometry of stirrers is irregular then a complete randomness of the EM field in all direction will be obtained.
An example of a reverberating enclosure is depicted in Figure 1, where S_1 and S_2 are the stirrers, T_x is a source for the power injected into the chamber and R_x is the receiving device which is free to move within the enclosure perimeter.
With the stirrers in a steady position, the field will be constant. The amplitude of the field received by an antenna positioned at *(x,y)* will depend on its position, as determined by the theory on resonant cavities. When stirrers rotate, the field will exhibit a variation that can be described by a statistical distribution which remains almost constant over the entire volume of the cavity. The field therefore appears homogeneous (at least, from a statistical point of view), and it is also de-polarized.

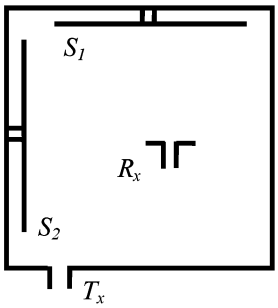


Figure 1 – Model of a Reverberating Enclosure

THE CHAOTIC MODEL

The main difference between the classical model and the one proposed by the author lies in the fact that isotropy and homogeneity of the field is obtained by making the geometry of the enclosure intrinsically chaotic so that each point of the test volume is hit by rays traveling every possible directions and with a uniform distribution of the incidence angles. In other words, mode mixing is obtained through chaotization of the trajectories rather than by a change in the boundary conditions.
A brief discussion about the choice made for the geometry is now in order. The reader is referred to (Fiumara et al. 1999; Fiumara et al. 2000) for further details.

The Chaotic Billiard

An introductory study about a new model for a reverberating enclosure is based on the analysis of the behavior of a point particle traveling at constant velocity inside a stadium-shaped billiard enclosure with perfect reflections at the boundary walls. To show that the behavior of incidence angles θ_i and incidence point θ_0 (see Figure 2) does not depend on the particular geometry chosen for the EUT but is intrinsic to the geometry of the enclosure, we present results obtained for both a perfectly reflecting and transparent object.

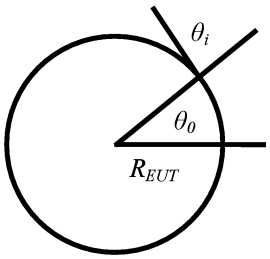


Figure 2 – Definition of Incidence Angles θ_i and Incidence Points θ_0

The simulation setup is presented in Figure 3. It shows that in both cases the dynamic appears to be chaotic since the point particle seems to cover the whole area within the billiard perimeter with all possible trajectories, i.e. traveling all possible directions.
Statistics for the quantities investigated are in Figure 4. They formalize the behavior that has been qualitatively shown in the preceding figure. The cumulative distribution of θ_i and θ_0 is indeed very close to be uniform, which expresses the property of isotropy of incidence in either EUT configurations.
The stadium-shaped billiard appears therefore to be the most suitable, since its chaotic behavior is independent of the nature of the object under investigation (Bunimovich and Sinai 1980).

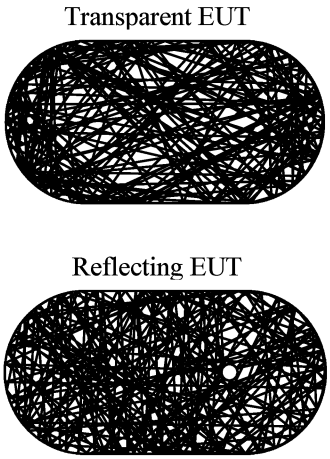


Figure 3 – Stadium-shaped Billiard with Transparent and Reflecting EUT

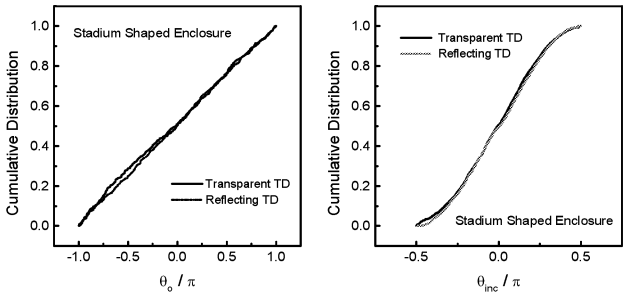


Figure 4 – Cumulative Distribution Functions (CDF) of θ_i and θ_0 for Transparent and Perfectly Reflecting EUT in a Stadium-shaped Billiard

The Electromagnetic Chaotic Billiard

We're now to move from the cinematic model used so far to investigate the properties of a chaotic billiard to a fully electromagnetic one upon which we can rely as a new model for an electromagnetic reverberation enclosure. It seems appropriate to use the chaotic billiard as the reference geometry. As for the EM equivalent of a point particle, an EM pulse will be used. Though, to maintain the hypothesis that the energy of the pulse remains localized in a small, virtually point region around the center of the pulse during propagation and after reflection against the walls, it is necessary to investigate the effect of the enclosure on the pulse characteristics. Those may vary from distortion of the wave front to a broadening of the longitudinal dimension which may render the pulse length comparable to the overall length of the enclosure, spoiling therefore its localized nature.

Pulsed Gaussian beam

A good representation of a localized energy carrier is obtained by shaping a Gaussian beam with a Gaussian time profile. A Gaussian beam is represented by the following expression:

$$G(x,z;b) = U_0\sqrt{\frac{jb}{z+jb}}e^{-jkz}e^{-j\frac{1}{2}k\frac{x^2}{R(z)}}e^{-\frac{x^2}{w^2(z)}}, \qquad (1)$$

where $w(z) = \sqrt{(2(z^2 + b^2)/kb)}$ is the *spot-size* and $R(z) = (z^2 + b^2)/z$ is the wave-front curvature. The minimum waist size, also known as *beam waist* is $w_x = 2w(0) = 2\sqrt{(2b/k)}$. The reader is referred to (Burkholder and Pathak 1991; Heyman and Felsen 2001) for further details about the Gaussian beam theory and applications.

From the expression given above it is apparent that the smaller the beam waist the larger the divergence in the beam. Therefore the need for a narrow beam strongly contrasts with the requirement of a beam that keeps collimated during propagation. Eventually, an optimal value for parameter $b \approx R_c$, where R_c is the radius of curvature of the side semi-circumferences, has been chosen to compensate the two contrasting behaviors.

The extension of the Gaussian beam that leads to a *pulsed* Gaussian beam has been obtained through an amplitude shaping along the longitudinal direction with a pulse, whose time duration determines the spectral content of the resulting pulse.

The profile has been chosen such that the frequency spectrum has a Gaussian shape itself. That is, an amplitude spectrum

$$P(\omega) = e^{\left(\frac{\omega-\omega_0}{\Delta\omega}\right)^2}$$

has been imposed, where $\omega_0 = 30\ GHz$ and $\Delta\omega = 5\ GHz$ are the central frequency and half-width frequency band used in our simulations. Values chosen limit the time extension to approximately $T_p = 2\cdot10^{-10}\ s$ along the propagation direction, as shown in Fig. 5.

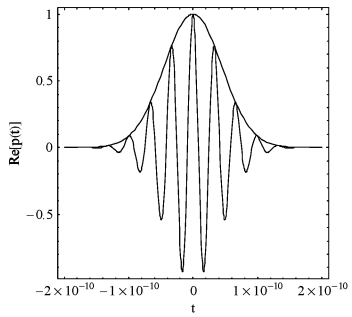


Figure 5 – Pulsed Gaussian Beam's Extension along the Direction of Propagation

Reflection of Pulsed Gaussian beam

Once injected into the enclosure, the Gaussian pulse propagates until reflection against one of the boundaries. Supposing a perfectly conducting wall and a non-dispersive medium, longitudinal profile remains unchanged after reflection. Not so for the transversal profile, where in principle different points in the wave front are differently affected by the curvature in the walls. It can be shown that (Burkholder and Pathak 1991):

$$U_0 = \Gamma(\theta_i)U_0'\sqrt{\frac{b'(\rho_c + jb)}{b(\rho_c' + jb')}}e^{-jk(\rho_c'-\rho_c)}$$

where U_0 is the amplitude and ρ_c is the distance between waist and point of incidence on the wall of the reflected pulse, and the primed quantities are relative to the incident pulse. Under the fundamental hypothesis of paraxial propagation and small incidence angles, we obtain the two relationships (see Figure 6):

$$\theta_r = \theta_i$$
$$\frac{1}{\rho_c + jb} = \frac{1}{\rho_c' + jb'} + \frac{2}{R_c\cos(\theta_i)} \qquad (2)$$

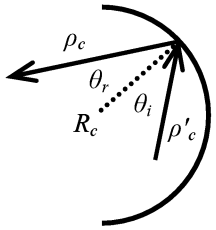


Figure 6 – Geometry of Incident and Reflected Pulse

Enclosure Configuration and Preliminary Results

After establishing the operating principles and constitutive relations for the new model we can now present the set-up chosen for the simulations carried out during the research.

With regard to the geometric configuration, we fixed: $a = R_c$, where a is the length of the straight wall of the enclosure; $\lambda_0 \ll D_{enc}$, where λ_0 is the wavelength corresponding to the central frequency and D_0 is the characteristic dimension of the enclosure, so that the pulse can be considered localized in its transverse dimension with respect to the chamber; $A_{EUT} \ll A_{enc}$ where A_{EUT} is the area of EUT – the shape used for the object under test is a circle, – and A_{enc} is the total enclosure area.

The electromagnetic characteristics have been fixed as follows: perfectly absorbing EUT, so that the incident pulse dissipates completely within the EUT without reflection; $T_p = 2\cdot10^{-10}\ s \ll 2a/c$, so that pulse's time duration is much less that the fly time between the two parallel walls. Pulses arise from a point in the wall and spread over an angular region of known extension. Source never radiates EUT directly though. In this manner statistics observed in the EUT are only affected by the enclosure and not by deterministic contributions by the source itself.

One last important comment involves the diverging behavior of the pulse inherited by its Gaussian nature. It may happen that at incidence onto EUT beam's spot-size is so large that the related power density does not cause any effect on the EUT, and will therefore be excluded when evaluating statistics from simulations. Since beam width has been discovered from preliminary studies to be strictly correlated to the number of reflections against curved surfaces of side walls – as already shown in similar situations

in (Palma 1996) – to keep the beam narrow we can think of limiting the maximum number of times that it can reflect on the circular walls before it is excluded from the simulations. As a matter of fact, limiting the number of reflections may pose a serious problem to the arising of chaotic behavior. To show that this is not the case, two pulses initially close to each other have been followed during propagation inside the billiard. After some $N = 6$ reflections distance was almost two orders of magnitude larger, and the Lyapunov exponent determining such evolution has been evaluated as:

$$\lambda_L = \frac{\ln 10^{\frac{\varepsilon_N}{\varepsilon_0}}}{N} \approx 0.7$$

where ε_0 and ε_N are the distances between rays at time $t = 0$ and after $N = 6$ reflections. The value obtained agrees with (Mackay 1988), showing that even if we limit the number of reflections to a maximum of $N = 6$ we can still claim the existence of chaotic dynamics.

RESULTS OF SIMULATIONS

Results presented in this section show the statistics obtained for the quantities θ_i and θ_0 as defined in Figure 2, in terms of Cumulative Distribution Functions (CDF). Reader is advised to pay closer attention to the range covered by the two quantities than the specific shape of the distributions. This because this preliminary research aims at evaluating the capability of the proposed model to cover the entire EUT with an incident pulse (i.e., uniformity) and to have each point hit by pulses coming from every direction (i.e., isotropy). Origin of the geometric axes has been positioned at the bottom left point of intersection between the circular and straight line walls.

Variation of EUT along x-axis

As EUT's position varies along the horizontal axis from the center of the enclosure (red curve in Figures 7 and 8) toward the right side wall (blue curve) the interval of values covered by θ_i and θ_0 start reducing and the extreme values $\theta_0 = 0$ and $\theta_0 = 2\pi$ disappear from plots. This is indeed expected, because as the room between EUT and right side wall decreases while the object moves toward the right side of the enclosure, less pulses will be able to penetrate in the narrow corridor and therefore that side of the EUT – in which $\theta_0 = 0$ and $\theta_0 = 2\pi$ are located – will be *shadowed* and not be hit by EM radiation.

At the same time the slope of plot increases and more pulses incident at $\theta_0 = \pi$ will be observed, though the point at which those maxima occur does not change significantly. The enclosure shows a higher focalization property of pulses that tend to impinge primarily onto the point that lies onto the symmetry axis of the chamber. Similarly, an increase in density around the incidence angle $\theta_i = \pi/4$ is noticed while negative angles are less numerous, but deeper investigation seems needed to understand the reasons for such change.

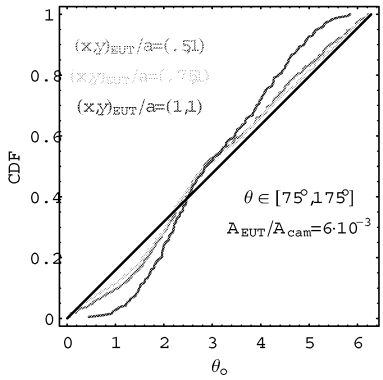


Figure 7 – CDF of θ_0 for different values of EUT's horizontal position

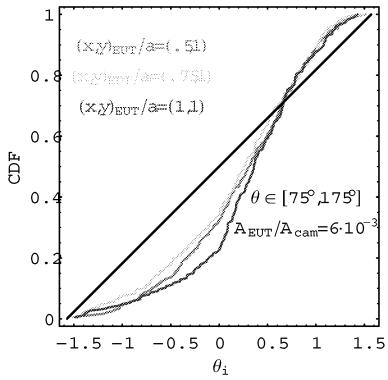


Figure 8 – CDF of θ_i for different values of EUT's horizontal position

Variation of EUT along y-axis

Variations of the y-position of EUT apparently imply stronger effects on angles' distributions than those observed for the x-axis – as Figures 9 and 10 show – even if in either cases variables appear to keep spanning over the whole range of possible values.

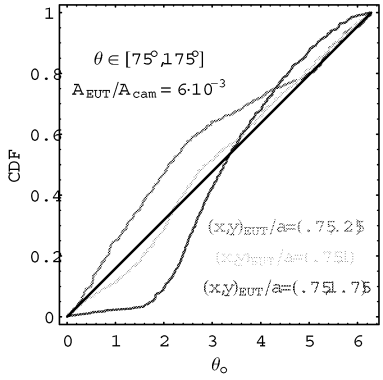


Figure 9 – CDF of θ_0 for different values of EUT's vertical position

That is, the distribution of values of θ_0 around $\theta_0 = 0$ reduce drastically as EUT approaches the upper side of enclosure (blue curve in Figure 9), and again this seems appropriate because of the same *shadow* effect seen before which covers those points of incidence. Cases where EUT is located very close to the upper or lower walls appear very different

from the uniform distribution observed when the object is located at the center of the enclosure. Maxima of θ_0's CDF move from $\theta_0 \approx \pi/2$ to $\theta_0 \approx \pi$ and higher, since points in the upper EUT's region become unreachable by pulses when moving toward the upper wall. On the other side, θ_i shows a variation from $\theta_i \approx$ -0.5 to $\theta_i \approx 0.5$ which seems appropriate because pulses will only be able to impinge onto the EUT from the right side of the normal axis to the incidence point.

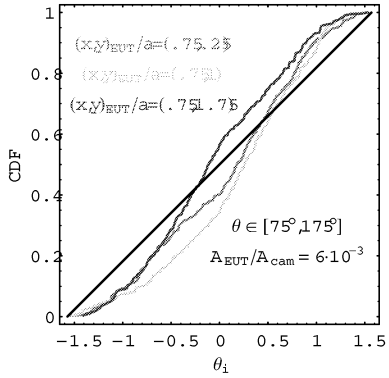


Figure 10 – CDF of θ_i for different Values of EUT's Vertical Position

Variation of EUT's radius

From Figure 11 we can observe that as the EUT increases in size, the density of θ_0 tends to peak around some point, which for the simulated scenario is $\theta_0 \approx 2.5$, and the interval itself tends to become smaller excluding point at its extremes. What we defined as a *shadow* effect again shows up, reducing the frequency of incident rays on the right side of EUT, as the EUT itself obstructs the path of pulses to the side of the enclosure.

On the other hand, Figure 12 shows that no remarkable differences arise in the distribution of θ_i for the values investigated.

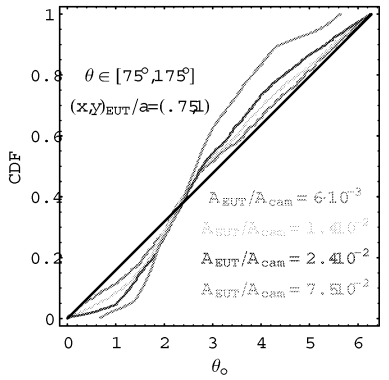


Figure 11 – CDF of θ_0 for different values of EUT's Radius

CONCLUSIONS

A new model for a reverberation enclosure has been presented in this paper, which merges the homogeneity and isotropy characteristics of a classical chamber to the wideband features, thus overcoming the intrinsic limit of a frequency domain (i.e., at single frequency steps) analysis of the electromagnetic immunity characteristics of EUT's.

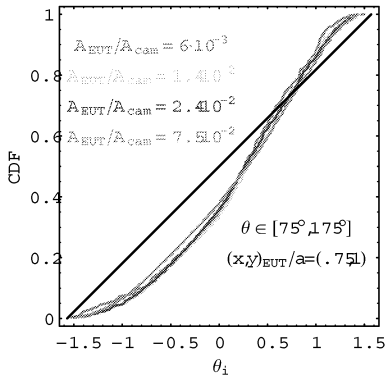


Figure 12 – CDF of θ_i for different values of EUT's Radius

The geometry has been chosen resorting to the principles of Chaos Theory, and simulations have shown that the enclosure exhibits a good behavior when the EUT is placed within an area centered in the center of the enclosure, not so large to reach the side walls. Optimal dimensions of the EUT are those that keep its perimeter not too close to the side walls. Future analysis will focus on the dependence of the proposed model's performance on different variables, such as geometric dimensions of the chamber and different EM parameters of the radiating pulse.

ACKNOWLEDGMENTS

The author wishes to thank Prof. Pierro and Prof Fiumara for providing useful hints during the execution of the preliminary theoretical study, and Prof. Angrisani for reviewing the paper and advising about future applications of the model.

REFERENCES

Bunimovich L.A. and Ya. G. Sinai 1980. *Comm. Math. Phys.*, Vol. 78, pp. 247

Burkholder R.J. and P.H. Pathak 1991. "Analysis of EM Penetration into and Scattering by Electrically Large Open Waveguide Cavities using Gaussian Beam Shooting," *Proc. IEEE*, Vol. 70, No. 10, pp. 1401-1412

Corona P.; G. Latmiral; E. Paolini; and L. Piccioli 1976. "Use of a Reverberating Enclosure for Measurements of Radiated Power in the Microwave Range," *IEEE Trans. Electromagnetic Compatibility*, Vol. 18, No 2, pp. 54-59

Corona P.; A. De Bonitatibus; G. Ferrara; and E. Paolini 1979. "Electromagnetic Enclosures Behaviour – comparison data analysis," *3rd Sump. Tech. Exhib. on EMC (Surich, Switzerland)*, pp. 231-236

Corona P. 1980. "Electromagnetic Reverberating Enclosures: Behaviour and Applications," *Alta Frequenza*, Vol. 49, pp. 154-158

Corona P.; G. Latmiral; and E. Paolini 1980a. "Performance and Analysis of a Reverberating Enclosure with Variable Geometry," *IEEE Trans. Electromagnetic Compatibility*, Vlo. 22, N. 1, pp. 2-5

Fiumara V.; V. Galdi; V. Pierro; and I.M. Pinto 1999. "From Mode-stirred Cavities to Electromagnetic Sinai Billards: Chaotic

Models of Reverberation Enclosures," *Proc. Int. Conf. Electromagnetics Advances Applications (ICEAA '99)*

Fiumara V.; V. Galdi; V. Pierro; and I.M. Pinto 2000. "Pulsed Reverberation Enclosures," *Proc. XIII RiNEm, Como Italy*, pp. 317-320

Hayman E. and L.B. Felsen 2001. "Gaussian Beam and Pulsed-beam Dynamics: Complex-source and Complex-spectrum Formulations within and beyond Paraxial Asymptotics," *JOSA A*, Vol. 18, No. 7, pp.1588-1611

Mackay A.J. 1988. "Chaos Theory and First-Order Ray Tracing in Ducts," *Electr. Lett.*, Vol. 34, No. 14, pp. 1388-1389

Palma C. 1996. "Complex Dynamics of a Beam in a Gaussian Cavity," *Optics Comm.*, No. 129, pp. 120-133

AUTHOR BIOGRAPHY

NICOLA PASQUINO, PhD, M'99 was born in Naples, Italy. In 1998 he graduated at the University of Naples, where he also received his Ph.D. in Electrical Engineering with a major in Electromagnetism. He currently is a post-doctoral research fellow in the Measurements Group of the University of Cassino (FR), Italy. He is also with the Department of Electrical Engineering of the University of Naples, Italy where he's involved in EMC compatibility, electromagnetic field measurements and quality assessment.

VALIDATION, ANALYSIS AND EVALUATION

A STATECHARTS-BASED METHODOLOGY FOR THE SIMULATION OF MOBILE AGENTS

Giancarlo Fortino and Wilma Russo
DEIS – Dipartimento di Elettronica, Informatica e Sistemistica
Università della Calabria
Via P. Bucci cubo 41C, Rende (CS)
Italy
E-mail: {g.fortino, w.russo}@unical.it

KEYWORDS
Distilled Statecharts, Mobile Agents, Java, Discrete-Event Simulation.

ABSTRACT

Mobile agent paradigm and technology are profitably being applied for the construction of a wide range of applications and systems in several areas encompassing e-business, telecommunications, meta-computing and military simulations. Such systems are usually complex so requiring methodologies and tools for an in-depth understanding of their dynamics and for their validation. This paper presents and exemplifies a methodology for modeling and validation through simulation of mobile agents based systems and applications. Mobile agents are modeled as asynchronous, event-driven single threaded entities whose behavior is specified using Distilled Statecharts. A distilled statechart, which embeds both the activity and the interactions a mobile agent performs during its lifecycle, can be easily translated into a Java composite object using the mobile active object framework. In order to analyze an agent-based system, a discrete-event, reflective simulation framework which allows for concurrency, timing and meta-level customization, is used. A case study is also provided which highlights the main features of the proposed methodology.

INTRODUCTION

Recently there has been a growing interest in designing and building distributed systems and applications in heterogeneous environments using the mobile agent paradigm and technology. Mobile agents are autonomous software components migrating within a network of execution environments to fulfill a goal-oriented task. They have been applied to several application areas ranging from telecommunications to e-business, from metacomputing to information retrieval over Internet (Gray et al. 2002). Such trials have shown the suitability and the potential benefits of mobile agents but, to date, their actual exploitation is rather limited. The main reason is that current mobile agent systems are still research prototypes so the majority of applications and systems developed using mobile agents can be developed using more traditional techniques. Indeed mobile agents allows for conservation of bandwidth, reduction in total completion time, reduction in latency, support for disconnected operation, support for load balancing and support for dynamic deployment, which are strengths uniquely shared only by mobile agents (Gray et al. 2002). In order to acquire a deeper understanding of the dynamics of mobile agent-based systems, applications, and basic technology before their actual development and deployment, simulation can be exploited. In particular, flexible simulation methodologies and tools are required which should be able to model, analyze, and validate:

(i) the activity (computations, interactions, and migrations) of mobile agents;
(ii) the basic mechanisms of the distributed architectures supporting mobile agents, namely mobile agent systems;
(iii) the behavior of applications and systems based on mobile agents.

Although several approaches and systems for the simulation of mobile agents are widely available (Uhrmacher et al. 2000; Theodoropoulos and Logan 1999), this research area is still young and prone to be further investigated.

In this paper, a methodology for flexible modeling and simulation of mobile agents is proposed. Modeling relies on Distilled Statecharts (DSCs) (Fortino et al. 2002) purposely derived from UML Statecharts (OMG 2001) which allow to specify the behavior of both mobile agents and stationary agents. Mobile agents are represented by asynchronous, event-driven and single-threaded entities which fulfill their goal-directed activity by alternating computations, interactions, and migrations. In particular computations and interactions are performed within a specific agent server (or location) whereas migrations are carried out across a network. Conversely stationary agents are not able to migrate and their main task is to grant secure access to the resources of their locations. At high level a complete agent-based system is modeled by a set of distilled statechart specifications associated either to mobile agents or to stationary agents. A DSC specification is translated into a composite object by means of the mobile active object framework, which is currently implemented in Java (Fortino et al. 2003). The simulation phase is enabled by a specialization of a discrete-event, reflective simulation framework exploited for the simulation of multimedia systems (Fortino and Nigro 2000) which embeds the composite objects obtained from a system specification by associating them to specific locations interconnected by a logical network. The simulation framework provides object execution through interleaving of timed and untimed events dispatching and meta-level customization by filtering of events generated by agents.

A REFERENCE MODEL FOR MOBILE AGENTS AND ITS IMPLEMENTATION

A mobile agent can be modeled as an event-driven lightweight agent and represented by the following tuple:

EA = <ID, DSM, CONTEXT, EQ, TC>,

where:

- ID is the unique identifier of the agent.
- DSM is a Distilled Statechart Machine object representing the dynamic agent behavior. Distilled StateCharts (DSCs) (Fortino et al. 2002) are a visual formalism derived from UML statecharts (OMG 2001) and purposely suited for event-driven single-threaded entities. A distilled statechart specifies the agent activity as a chain of atomic actions executed during specific state transitions triggered by internal and external stimuli (or events).
- CONTEXT is the basic agent class encompassing specific agent methods, references to the execution environment, and (possibly) data. In particular, it provides the primitive *generate(<eventname>(<parameters>))*, which allows to asynchronously generate self-triggering events or interaction events targeting other agents.
- EQ is the event queue buffering the received events.
- TC is the thread of control supporting the agent execution.

In particular, an agent computation can be defined in terms of processing of the events serially extracted from its queue. An agent computation step, which is cyclically repeated till the agent termination, consists of: (i) removing an event, if any, from the event queue; (ii) reacting to the event according to its dynamic behavior; (iii) (possibly) changing its local state. During (ii), an agent can generate zero, one or more events.

Agent migration

A mobile agent initiates a migration operation by generating the *Move(Destination)* event targeting itself for autonomous migration or targeting another agent for passive migration. In both cases, the actual migration can occur only after receiving and processing the *Move* event. In order to enable transparent migration according to a coarse-grained strong mobility model (Karnik and Tripathi 1998), the agent DSM is based on the agent mobility template (Fig. 1).

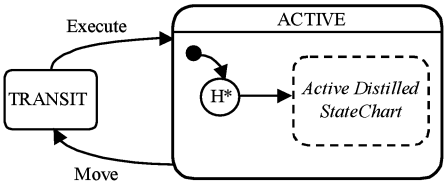


Figure 1: The Agent Mobility Template.

Such a template assures that every time the agent migrates, i.e., it goes from the Active to the Transit state, and then it is restored at the destination site, i.e., it turns back from the Transit to the Active state, the agent execution is restored exactly in the same state it left off before migrating. In addition the agent mobility is of coarse-grained strong type since the agent can migrate only at specific points of its execution, i.e., after completing the reaction to an event.

Object-oriented design and implementation of event-driven lightweight agents

The MAO (Mobile Active Object) framework (Fig. 2), which is purposely derived from the Statecharts Pattern (Yacoub and Ammar 1998) and currently implemented in Java, is used to translate the DSC specification of a dynamic agent behavior first into a design object and then into an executable composite object (Harel and Gery 1997).

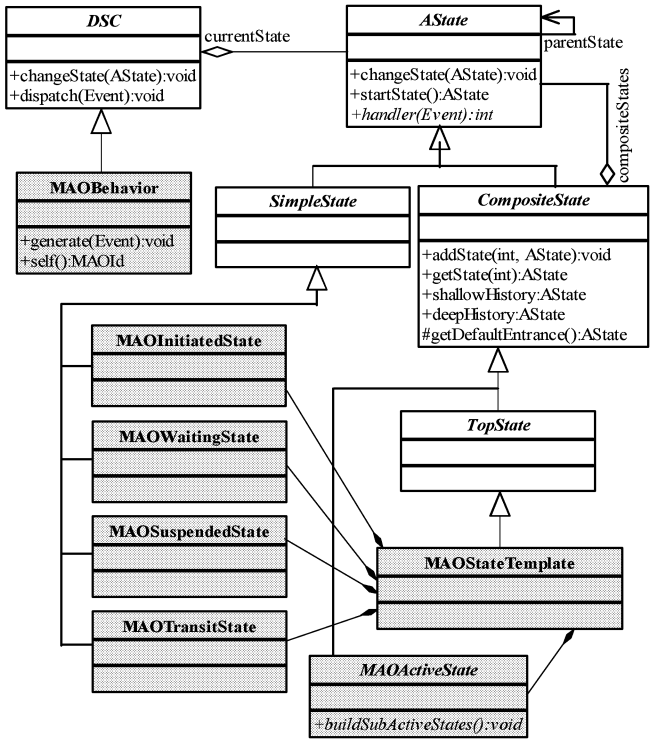


Figure 2. The UML class diagram of the MAO Framework

The MAO framework (Fortino et al. 2003) defines a FIPA-compliant architecture (FIPA 2002) for the dynamic behavior of lightweight agents and provides the basic building block to create them. Its frozen spots define the overall architecture of a MAO characterized by classes which remain unchanged in any instantiation of the framework (concrete classes in Figure 2), whereas its hot spots are to be adapted to the needs of the specific agent under development. To achieve adaptability it is necessary to subclass the abstract class `MAOActiveState` in order to define the specific active behavior of the MAO. In addition the following rules should be followed in order to design and implement the Active Behavior DSC:

- the classes of the specific substates transitively nested in the active state should extend either `SimpleState` or `CompositeState`
- the introduced substate classes should be linked, using a composition association, to one another and to the specific active state class according to their hierarchical structure

- state variables should be declared in state classes at a certain level of the hierarchy according to their scope
- events labeling transitions of the visual specification are classes extending `MAOEvent`
- action expressions and guards should be enclosed in methods
- transitions are masked inside the method handler of each specific state. In particular, a transition originating in a given state should be realized within the handler of that state.

Thus, a `MAOBehavior` object is a composite object encapsulating the dynamic behavior of an agent. It is activated through the following steps:

(i) its constructor receives the specific `MAOActiveState` object, creates the `MAOStateTemplate` object and passes it the reference to the specific active state.

(ii) the start method is invoked on the `MAOBehavior` object, which sets its current state to `MAOInitiatedState`.

A generic event-driven lightweight agent (`GenericAgent`) is obtained by associating and adapting a `MAOBehavior` object to an agent class further encompassing an identity object, a queue object, and generally a thread. Figure 3 shows a UML class diagram of the agent adaptation framework. The run method cyclically extracts the next available event to be processed from `MAOQueue` until `MAOBehavior` reaches the final state. A dequeued event drives `MAOBehavior` through the dispatch method. The receive method is invoked either by the execution environment (or agent server) or by the agent itself when an event targeting a `GenericAgent` is generated. The generate method directly enqueues a `MAOEvent` if it is a self-trigger, otherwise uses the specific coordination mechanisms of `GenericAgent` to route the event toward its final destination.

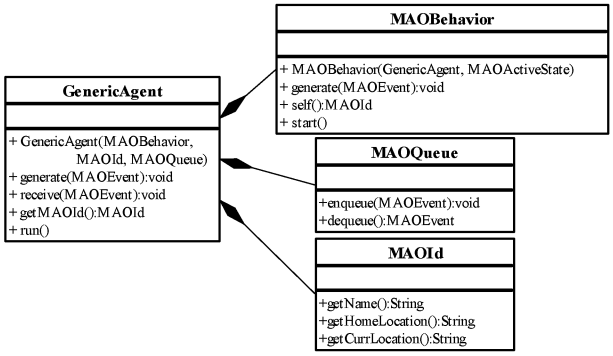


Figure 3. The Agent Adaptation Framework

Although the goal of this paper is mainly oriented to mobile agents simulation, the obtained event-driven lightweight agent is independent from any real or simulation context and can seamlessly be adapted to real mobile agent systems. In particular, it can be either supported by an ad-hoc built agent platform as in (Fortino et al. 2002) or mapped onto any mobile agent or object system for real executions. In fact, every currently available mobile agent system (Gray et al. 2002) provides a specific mobile agent programming abstraction along with coordination primitives, naming, and other facilities through which a mobile agent can be actually implemented.

A DISCRETE-EVENT, REFLECTIVE SIMULATION FRAMEWORK

A mobile agent-based system or application can be seen as a set of stationary or mobile event-driven lightweight agents, which are obtained using the approach described in the previous section. In order to simulate them, agents are embedded into a Java-based simulation framework which basically provides: (i) execution of agents by interleaving their event processing, (ii) interchange of events among agents, (iii) migration of agents, and (iv) clustering of agents into networked locations.

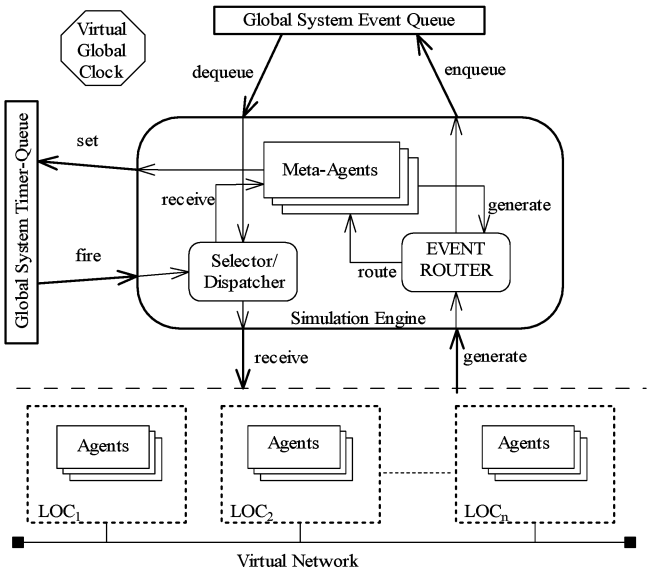


Figure 4. The Simulation Framework Architecture

The architecture of the framework is reported in Figure 4. Its basic components are:

- *Global System Event Queue (GSEQ)*, which stores all the untimed system events to be delivered. An event is an instance of `MAOEvent` which contains the `MAOIds` of the event source and target, and (possibly) data objects.
- *Global System Timer Queue (GSTQ)*, which stores all the system timers. A timer is an object encapsulating a `MAOEvent` and a timeout. The event is delivered at the timeout expiration. The timeout refers to a virtual simulation time.
- *Virtual Global Clock (VGC)*, which represents the virtual global time under the hypothesis of clock synchronization among networked nodes. It is incremented every time that a timer expires.
- *Simulation Engine (SE)*, which drives the execution of a simulation. It is further composed of:
  - a set of *Meta-Agents*, which are stationary agents installed on a group of agents which filter their event generations. When Meta-Agents act as *Synchronizers* (Fortino and Nigro 2000), they can capture an event and schedule its actual delivering to occur after a timeout by setting a Timer.
  - *Event Router (ER)*, which receives the events generated by the agents and insert them into GSEQ if they are not subjected to meta-level customization; otherwise ER routes the events to their associated meta-agents.

- *Selector & Dispatcher (SD)*, which cyclically extracts an untimed event from GSEQ and dispatches it to the target agent. If no events are present in GSEQ, SD forces a timer (the one with the lowest timeout) to fire and dispatches the associated event to its target.

- *Agents*, which are the computational entities, stationary or mobile, of an agent-based system. They can only perform computational tasks, i.e., computations, interactions and migrations. Agents are stored into the global system white pages table (GSWP) whose entry is a tuple <`MAOId`, `GenericAgent`>. An agent is an instance of `SimulationGenericAgent` which adapts `GenericAgent` to the simulation environment. It contains a reference to SE which supports the agent execution and makes it available the `generate` method. The `run` method and the `MAOQueue` are disabled.
- *Virtual Network*, which is the virtual network infrastructure, realized by means of specific meta-agents, connecting the *Virtual Agent Locations (VALs)*. VAL is implemented as a symbolic name which can be referred by the *home location* and *current location* attributes of the `MAOId` of an agent.

Meta-level Customizations: Network Events and Migration

Meta-level customization allows for separation of concerns: computational agents are unaware of network and timing issues and only perform functional tasks whereas meta-agents impose timing constraints to generated events according to network and system dependent parameters (Fortino and Nigro 2000).

Network events are events whose source and target agents are on different VALs. Their transmission is supported by the Network Meta Agent (NMA) which captures and constrains them to occur after a timeout set according to network parameters (e.g., link delay) and event size.

The agent migration process supported by the Migration Meta Agent (MMA) involves the following steps:

1. MMA captures the Execute event, generated as a consequence of the transition from Active to Transit, and creates a timer, containing the event itself, whose timeout is set to the instant in time of migration completion according to network connection parameters and agent dimension (code and data).
2. MMA sets the current location to the new agent location in the `MAOId` of the migrating agent. Events delivered to an agent during its migration, if local to the new agent destination, are buffered into a temporary event queue (TEQ) associated to the migrating agent; otherwise, events are handled as network events and directed through a forwarder (FW) to the future agent location.
3. After the timer expiration, the Execute event is delivered to the agent and its processing restores the agent back into the Active state and inserts all the events contained in TEQ into GSEQ.

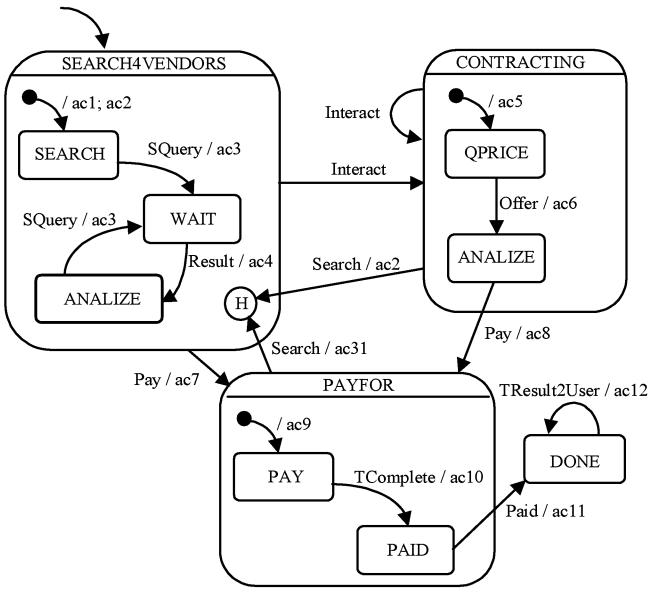
Both the NMA and MMA refer to a specific-format file which contain the network topology description. By default, the network topology is modeled as a completely connected peer-to-peer network of VALs, even though other topologies can be easily constructed.

A CASE STUDY

The case study refers to the analysis of the dynamics and strategies of an agent-based electronic market (Bredin et al. 1998). The agents of the electronic market are:

- *Traveling SalesAgent (TSA)*, which represents a traveling salesperson whose goal is to look for a specific product that meets the needs of its customer. It moves across the network to locate all the vendors of the required product by interacting with a distributed system of yellow pages (YP) which offers the catalog of vendors. After retrieving a list of vendors, TSA moves to the vendor sites in order to buy the most convenient product. In order to pay for the YP service and to buy a product, TSA is equipped with a wallet containing a limited amount of e-cash.
- *YellowPage Agent (YPA)*, which represents an entry point of the YP distributed service. YPA is queried by TSA for information on vendors of specific products. YPA returns either the requested info or the location of another YPA. Each query costs a fixed amount of e-cash.
- *Vendor Agent (VA)*, which represents the vendor of specific products. In order to buy a specific product, TSA asks VA for the product price and decides to buy it or to move to another VA location.
- *Bank Agent (BA)*, which represents a reference bank of TSA, VA, or YPA. Bills owned by such agents are unique, cryptographically signed documents issued by one of the accredited banks (set of BA). A bill is a few bytes of information containing: the name of the bank (or BA), the amount of the bill, the unique bill identifier, and the bank's signature needed to check the authenticity of the bill. When TSA buys a specific product or submits a searching query, it has to pay VA or YPA for a given amount of bills. Under the hypothesis that each bill has the same value and that each product price is equal to an integer number of bills, the money transaction between two agents (Buyer and Seller) runs as follows: Buyer gives the bills to Seller which adds the bills to its wallet whereas Buyer removes them from its wallet. The bills should be validated by Seller in order to prevent their duplicated use by Buyer.

The active dynamic behavior of TSA is reported in figure 5. In the SEARCH4VENDORS state, TSA searches for vendors by navigating across the yellow pages distributed system. Before submitting the queryvendor to a YPA, TSA pays 1 bill for the requested service in the PAYFOR state. Afterwards, it analyzes the obtained result using the `enoughVendors` method. If TSA has collected a sufficient number of vendor identifiers or no more available YPAs exist, TSA moves to the chosen next vendor location for contracting with a VA. In the CONTRACTING state, TSA first asks VA for the product price; then, on the basis of the evaluation of the received offer made by the `price_is_ok` method, it can decide to buy the product or to move to the next VA or to search for new VAs. When TSA purchases a product, it comes back to its home location and notifies the transaction result to its owner. Table 1 reports, for sake of clarity, the meaning of state variables, methods and interaction events of Figure 5. Moreover, for sake of simplicity, Figure 5 does not highlight the exceptional cases of unsuccessful purchase.




```
ac1 :int nextYP=0;
ac2 :YPAtarget=(MAOId)yp.elementAt(nextYP++);
     generate(new MAOMove(self(),YPAtarget.getCurrLocation()))
     generate(new Pay(self()));
ac3 :generate(new QueryVendor(self(),YPAtarget,queryvendor));
ac31:generate(new SQuery(self()));
ac4 :Result r = (Result)e;
     if (r.getAvailableVendors()) {
       addVendors(r.getVA());
       ac41();
      else { addYPA(r.getYPA()); ac2(); }
ac41:if (enoughVendors()) {
      VAtarget=(MAOId)va.getNext();
      generate(new MAOMove(self(),VAtarget.getCurrLocation()))
      generate(new Interact(self()));
     }
     else ac2();
ac5 :generate(new QueryPrice(self(),VATarget,queryprice));
ac6 :Offer o = (Offer)e; prodcost = o.getPrice();
     if (price_is_ok(prodcost)) generate(new Pay(self()));
     else
     if (enoughVendors()) {
      VAtarget=(MAOId)va.getNext();
      generate(new MAOMove(self(),VAtarget.getCurrLocation()))
      generate(new Interact(self()));
     }
     else (generate(new Search(self()));
ac7 :amount = 1; bills = prepareBills(amount); ypflag=true;
     target = YPATarget;
ac8 :amount = prodcost; bills = prepareBills(amount);
     ypflag=false; target = VATarget;
ac9 :generate(new PayFor(self(), target, bills);
ac10:nbills = nbills - amount;
     if (ypflag) generate(new Search(self()));
     else generate(new Paid(self()));
ac11:generate(new MAOMove(self(),self().getHomeLocation()));
     generate(new TResult2User(self()));
ac12:reportTR();
```

Figure 5. The Active DSC of TSA

The strategies that TSA uses to search for and buy products are embedded basically in its dynamic behavior and specifically in the methods `enoughVendors` and `price_is_ok`. It is worth noting that TSA buys a product if and only if the product price is equal to or less than P_{max}. The implemented strategies are the following:

- *Minimum Price*. TSA first searches for all the possible VAs providing a specific product and, then, interacts with all the found VAs to find out the best price for the product. Finally, it moves to the location of the VA which offered the best price and buys the product.
- *First-shot*. TSA first searches for VAs providing a specific product and then visits such VAs one by one. As soon as a VA offers an acceptable price (i.e., <=P_{max}), TSA buys the product.
- *Fixed-trials*. TSA first searches for VAs providing a specific product and then visits such VAs one by one. After a fixed number of trials (or received offers), TSA moves to the location of the VA which offered the best price and buys the product.
- *Random-trials*. Similar to fixed-trials but using a random number of trials uniformly distributed between a minimum and a maximum number.

Table 1. State Variables, Methods and Events related to TSA

VAR	DESCRIPTION
yp	the list dynamically constructed of YPA identifiers
va	the list dynamically constructed of VA identifiers
nextYP	the number of visited YPAs
VATarget	the current VA to be contacted
YPATarget	the current YPA to be contacted
prodcost	the price of the product returned by a VA
amout	the number of bills to be paid
nbills	the total number of bills got by TSA in its wallet
bills	the array of bills to be transferred
ypflag	a boolean indicating payment of YP service or not
e	the reference to the triggering event instance received
EVENT	**DESCRIPTION**
Result	the event, sent from YPA, containing the queryvendor result (a list of VA ids or a list of YPA ids)
Offer	the event, sent from VA, containing the requested product price
TComplete	the event, sent from VA or YPA, which confirms that the payment transaction has been completed
PayFor	the event, sent from TSA to either YPA or VA, containing the payment bills
QueryPrice	the event, sent from TSA to VA, requesting a specific product offer
QueryVendor	the event, sent from TSA to YPA, requesting a list of vendors of a specific product
METHOD	**DESCRIPTION**
prepareBill	the method packetizes the right amount of bills in an array
reportTR	the method notifies the TSA's owner about the transaction result

A Simulation Scenario

A basic simulation scenario to preliminary analyze the dynamics of the agent-based market is designed as follows:

- TSAs are created according to an exponential distribution on a set of VALs corresponding to the customers' locations. A TSA is fed with an initial YPA identifier, the product identifier, the maximum price (P_{max}), and the wallet.
- YPAs, VAs, and BAs are associated to three disjoint sets of different VALs.
- A fixed number of product types ($N_p=N_b+N_s$) is sold in the market, where N_b is the number of the basic products sold by all VAs, and N_s is the number of specific products sold by a subset of VAs. The quantity of each product is not limited.
- YPAs have only a static partial list of products associated to VAs. Moreover they have references at least to one YPA.
- VAs offer a partial list of different products which always contains the basic products. The price of a product, which is uniformly distributed between a minimum and a maximum price, is set in each VA at initialization time and is never changed. Thus VAs adopt a fixed-pricing policy to sell products.

- The virtual network connecting all the VALs is based on the default network topology whose links have the same mean delay. Moreover, event size and agent dimension are not taken into consideration so not affecting event transmission and migration respectively.

The goal of the simulation is to:

1. validate the activity (computations, interactions, and migrations) of TSA, and that of the stationary agents.
2. analyze the behavior of the agent-based electronic market with respect to the different strategies adopted by TSA.
3. evaluate both the basic mechanisms (e.g., event forwarding and migration) of a general mobile agent system and the migration costs versus message passing or remote method invocation on the basis of a performance model for mobile agents (Strasser and Schwehm 1997).

The study currently performed validates the activity of the agents and shows the liveness of the agent-based e-market. More in depth analyses are being performed aiming at quantitatively evaluating the TSA's strategies. For instance a performance measurement is the average time spent to fulfill the purchase task out of the total number of bills paid for searching and buying the product.

RELATED WORK

JAMES (Java-based Agent Modeling Environment for Simulation) (Uhrmacher et al. 2000) is a modeling and simulation framework for agent-based complex systems based on parallel DEVS. Agents as well as mobile agents are described as atomic models integrated into coupled models representing locations. Network connections are also modeled as atomic models called channels. Simulation is performed using the DEVS abstract simulator model. JAMES is currently the most cited framework for agent-based simulation which also embeds simulation of mobile agents. Moreover, other simulation environments for mobile agents are available and allow to: (i) explore how mobile agents can be used to update the information available in a static network and what role collaboration between agents plays in this context (Minar and Kramer 1998); (ii) analyze the market-based resource control system of D'Agents and, in particular, the resource allocation mechanism of a resource manager using a sealed-bid second-price auction policy (Bredin et al. 1998). Such environments are built from scratch and are not based on a well-defined methodology as JAMES and the approach proposed in this paper.

CONCLUSION

Flexible methodologies and tools for the simulation of agent-based systems are highly required to effectively support agent-oriented software development. This paper has proposed and exemplified a Statecharts-based methodology which supports modeling and simulation of mobile agent-based systems and applications.

The novelty of the proposed approach is twofold: (i) the agent modeling and implementation can be seamlessly adapted to both real implementations and simulations; (ii) the simulation framework allows for meta-level customization which further augments the simulation flexibility.

In order to exemplify the proposed methodology the modeling and the simulation of an agent-based electronic market has been described. The case study can fit both an e-commerce system and a computational resource control system where the vendor agents are the resource managers and the products are memory pages, CPU cycles, storage space, database query and so forth. A basic simulation scenario is designed which allows for validation of agents and for a preliminary study of the dynamics (e.g., liveness) of the agent-based electronic market.

On going work is geared at (i) quantitatively analyzing the traveling salesagent's strategies and (ii) introducing dynamic-pricing strategies for vendor agents.

REFERENCES

Bredin, J.; D. Kotz; and D. Rus. 1998. "Market-based Resource Control for Mobile Agents." In *Proc. of ACM Autonomous Agents* (May).

FIPA. 2002. *FIPA Agent Management Support for Mobility*. Specification DC00087C, 2002/05/10. http://www.fipa.org.

Fortino, G.; W. Russo; and E. Zimeo. 2003. "A Farmework for Design and Implementation of Mobile Active Objects". In *Proc. of IASTED Software Engineering and Applications* (Marina del Rey, CA, Nov 3-5).

Fortino, G.; F. Frattolillo; W. Russo, and E. Zimeo. 2002. "Mobile Active Objects for highly dynamic distributed computing." In *Proc. of IEEE IPDPS'02, Workshop on Java for Parallel and Distributed Computing*, (Fort Lauderdale, USA, Apr).

Fortino, G. and L. Nigro. 2000. "A Methodology Centered on Modularization of QoS Constraints for the Development and Performance Evaluation of Multimedia Systems." In *Proc. of 33rd Annual Simulation Symposium* (Washington DC, Apr).

Gray, R.; D. Kotz; G. Cybenko; and D. Rus. 2002. "Mobile Agents: Motivations and state of the art systems.", in *Handbook of Agent Technology,* J. Bradshaw (Ed.). AAAI/MIT Press. to appear.

Harel D. and E. Gery. 1997. "Executable Object Modeling with Statecharts." *IEEE Computer*, (July), 31-42.

Object Management Group. 2001. *Unified Modeling Language*. Specification N. formal/01-09-67, v.1.4, (Sep). http://www.omg.org.

Karnik, N.M. and A.R. Tripathi. 1998. "Design Issues in Mobile-Agent Programming Systems." *IEEE Concurrency*, (Jul/Sept).

Minar, N.; K.H. Kramer; and P. Maes. 1998. "Cooperating Mobile Agents for Dynamic Network Routing." In *Proc. of the 1st Hungarian National Conference on Agent Based Computation.*

Strasser, M. and M. Schwehm. 1997. "A Performance Model for Mobile Agent Systems." In *Proc. of PDPTA*, CSREA, 1132-1140.

Theodoropoulos, G. and B. Logan. 1999. "A framework for the distributed simulation of agent-based systems." In *Proc. of the 13th SCS European Simulation Multiconference*, 58-65.

Uhrmacher, A.M.; P. Tyschler; and D. Tyschler. 2000. "Modeling and Simulation of Mobile Agents." *Future Generation Computing Systems* 17, 107-118.

Yacoub, S.M. and H.H. Ammar. 1998. "A Pattern language of Statecharts." In *Proc. of the PloP-98* (Illinois, USA, Aug).

SIMULATING HUMAN RESOURCES IN SOFTWARE DEVELOPMENT PROCESSES

Thomas Hanne
Fraunhofer Institute for Industrial Mathematics (ITWM)
Department of Optimization
Gottlieb-Daimler-Str. 49, 67663 Kaiserslautern, Germany
E-mail: hanne@itwm.fhg.de

Holger Neu
Fraunhofer Institute for Experimental Software Engineering (IESE)
Department for Quality and Process Engineering
Sauerwiesen 6, 67661 Kaiserslautern, Germany
E-mail: neu@iese.fhg.de

KEYWORDS

Human resource modelling, software process, simulation, productivity, human factors, human resource management.

ABSTRACT

In this paper, we discuss approaches related to the explicit modelling of human beings in software processes. While in most older simulation models of software processes, esp. those of the system dynamics type, humans are only represented as a labor pool, more recent models of the discrete-event simulation type require representation of individual humans. In that case, particularities regarding the person become more relevant. These individual effects are either considered as stochastic variations of productivity, or an explanation is sought based on individual characteristics, such as skills for instance. In this paper, various specific models for representing human effects in software process simulation are discussed.

INTRODUCTION

During the last few years, simulation has become an important approach for modelling and analyzing software development processes (Kellner et al. 1999). Most of the models discussed in the literature are of the system dynamics type, which is well-suited for representing and utilizing the cause-effect-relationships between various variables related to software development processes. Typically, such models are based on aggregated variables such as, for instance, total workforce, total amount of work, or total number of defects.

Such a level of abstraction is appropriate for a rough estimation of overall project variables such as duration, effort and costs, but does not solve typical tasks of a project manager such as deciding on process alternatives (e.g., inspecting a design or code document, testing a module), assigning tasks to persons, or scheduling the various activities. Simulation models with an explicit representation of objects related to software development processes, i.e. items (design or code documents) and developers, might be better suited for supporting these decisions. Typically, discrete-event simulation models allow for such an explicit representation of a multiplicity of objects of the same type.

In two projects (project SEV and project ProSim) we dealt with the development of discrete-event simulation models of software development processes (Neu et al. 2002, 2003). In those projects, we faced various problems related to the rather detailed level of modelling, especially problems related to an adequate representation of persons (developers) participating in the processes. When considering human-based models of software development processes we found that few established approaches exist and that some models appear to have an ad hoc character (mainly because of a lack of empirical data available for validating models). For these reasons we consulted empirical results from psychology and other fields and tried to employ reasonable (but still simple) models for considering the most important effects related to software developers. Due to lack of space, the basic empirical findings are discussed in a subsequent paper. Here, we focus on human modelling as used in the discrete-event simulation. Other aspects of the discrete-event simulation model such as the modelling of artefacts (code documents), the representation of the activities coding, inspection, test, and rework, and the organization of the overall process (assignment of staff, scheduling) are not treated in this paper, but can be found in (Neu et al. 2002, 2003, Hanne and Nickel 2003).

The paper is organized as follows: In the following section, related work on modelling human factors in software development processes is discussed. Then, some of our approaches to human performance modelling in discrete-event simulation are presented. With respect to this model scope and available or obtainable data, we concentrate on models for learning and time pressure effects. After that, aspects of the overall simulation model and the utilization of stochastic elements are discussed. The paper ends with the conclusions.

RELATED WORK

Several authors describe human resource management or models for human resources in software process models.

These approaches can be distinguished into two categories; first, models that treat the participants in the model as one or two groups of equal persons, and, second, individual modelling of each participant in the process. The first approach is common in system dynamics models because individual modelling of the persons would lead to a complicated model. The second approach is realized in models with different simulation methods, e.g., discrete-event models or generalized stochastic Petri-nets.

(Abdel-Hamid and Madnick 1991) propose a model for human resource management as part of their system dynamics model. They divide the workforce into two groups, the experienced and the newly hired (inexperienced) people, with some of the productive time of the experienced personnel being needed to train the inexperienced personnel. Hiring new personnel, removing personnel (by management to other projects) and people quitting are modelled with delays. The learning curve for becoming an experienced person is modelled by an assimilation rate that describes the time it takes to acquire the necessary knowledge. The curve is modelled as a first order exponential delay.

A similar model, called two level promotion chain, is proposed in (Sterman 2000). This model is for a more general use to demonstrate a worker training with two levels of experience. This model can be easily extended to more than two experience levels, each with a different learning curve between two levels.

(Rus 1998) incorporates a learning factor into the production activity and the validation & verification activity block. The learning factor is modelled as an (unexplained) graphic function and is multiplied with the average productivity to get the actual productivity.

In the approaches above, the participants in the process are treated equally. It is not possible to model individual abilities or learning curve parameters. This behaviour is sufficient in models that show a more generalized or upper management view of the process. In models that need to show a more detailed picture, e.g., for scheduling or decisions based on individual attributes, this is not sufficient.

A more detailed view on the learning curve in software development process simulation is given in (Hanakawa et al. 1998). They propose three sub-models for modelling the learning curve. The activity model captures the type and difficulty of the tasks to be done. The activity model is a distribution of knowledge levels that were derived from the amount of activities that have the same knowledge level. The productivity model shows the productivity of the developer in the context of his or her individual knowledge and the knowledge level needed for a specific task. The knowledge model depicts how much knowledge the developer gains by executing a task where a specific knowledge level is needed. The simulation model has seven steps plus step 0 for initialisation. An activity is selected (Step 1), then productivity is computed (Step 2). The accomplished work from the activity model is subtracted (Step 3) and the quantity of gained knowledge is calculated (Step 4). According to the gained knowledge, the knowledge level is updated (Step 5) and the productivity level in the productivity model is updated (Step 6). Depending on the remaining activities, the model ends or starts for the next time increment at the beginning (Step 7).

The authors are conducting several case studies for predicting development time under different settings for the developers. They also show the increase of the developers' knowledge (learning curve) during the simulation. Besides the parameters of the knowledge model, the selection policy for the activities to be accomplished and the shape of the activity model (distribution) have great impact on the learning curve. With normal distribution in the activity model and a random selection policy the learning curve is similar to the one we propose in Figure 2.

In (Hanakawa et al. 2002) the authors use the same model as in (Hanakawa et al. 1998). They add cognitive maps to describe the structure of individual knowledge of the developers. The same maps are used to relate the project workload to certain areas of knowledge. With this addition, the developers' level of knowledge can be simulated for different areas in the cognitive map.

(Christie and Staley 2000) are modelling issues associated with the social dynamics of requirements development. The authors use social skills for describing people in a discrete-event simulation model for simulating teams creating requirement documents. Three skills are modelled to describe the technical and domain experts: technical capability, ability to convince (influence), and considering ideas of others (openness).

The skills of the supporting roles (leader, facilitator, scribe) include enabling agreement and modifying the skills of the technical personnel. These characteristics are modelled using a real uniform distribution in the range of 0.0 to 1+x, where x takes on a small positive value. Even if a learning curve is not explicitly modelled, the constant value x has the effect of a learning curve because it increases understanding during the sessions and leads to a better quality of the requirement documents.

A VARIETY OF SKILLS

A key assumption of our model is that specific skills determine personal productivities (i.e., the speed of work) on the one hand, and, on the other hand, the quality of work, for instance, the produced number of defects during coding or the found number of defects during inspection and testing. Since productivity-oriented and quality-oriented skills do not need to be the same for a person or evolve synchronously, we distinguish between these two types of skill values in the model (see bottom level of Fig. 4). For instance, the specific skill values for a developer j of the model are the coding quality skill (cqs_j), the preparation (inspection) quality skill (pqs_j), the testing quality skill

(tqs_j), the coding productivity skill (cps_j), and the preparation productivity skill (pps_j).

For ease of use we assume these skill values to be calibrated on a nondimensional [0,1] interval. A skill value of about 0.5 is typical for an average developer, while a value near 1 characterizes an experienced or high performance developer. Multiplied with given values for a maximum quality of work or a maximum productivity (corresponding to skill values of 1), a person's actual defect (production, detection) rate and productivity can be determined. Thus, the following model parameters with respect to productivities are used: maximum coding productivity, *mcp*, maximum preparation productivity, *mpp*, and maximum testing productivity, *mtp.* With respect to defects, the following model parameters are used: The number of defects in relation to the size of the document to be produced is expressed by minimum defect density, *mdd*.

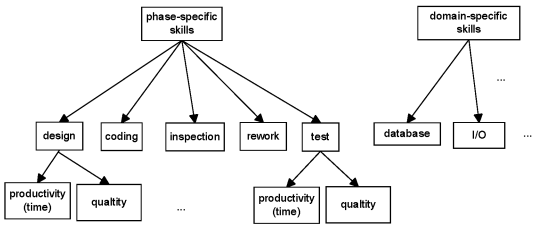


Fig. 1: Structure of Skills

LEARNING

In the simulation model, the skill values of software developers are dynamically varied by employing a learning model. The learning mechanism is based on a logistic growth model. According to the pure logistic model, a minimum value of 0 and a maximum value or 1 is assumed with a continuous monotonic growth of the corresponding skill output variable (see Fig. 2).

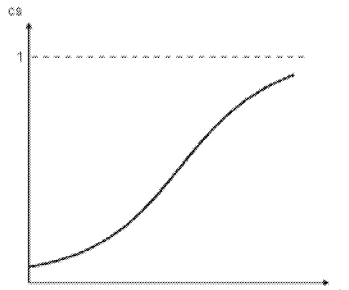


Fig. 2: Logistic Growth of Skills

Assuming small changes of time, Δt, corresponding changes of the skill value can be described by

$$\Delta skill := lf \; \Delta t \; skill \; (1 - skill)$$

where *lf* is a learning factor specific to the particular skill, which depends on the institutional environment and personal factors not explicitly considered in the simulation model. Δt is the time used for an activity that involves the particular skill, thus the time elapsed since the last update of that skill. If an activity involves several skills (e.g., phase-specific and domain-specific skills), all of them should be updated after finishing the activity. The factor *lf* determines the steepness of the learning curve and thus the time needed to get from a low skill value to a high skill value. Therefore, it reflects the average learning capability of a software developing organization and depends, for instance, on the degree of re-usability of former artefacts.

TIME PRESSURE, FATIGUE AND BOREDOM

For taking another main effect on personal productivities into account, we have developed a time pressure model for capturing some motivational effects related to fatigue and boredom. Considering a standard level of workload and time pressure, it is assumed that an increase of time pressure first leads to an increase of productivity. For instance, people work more concentrated, social activities are reduced or voluntary overtime is done. This, however, can usually only be expected for a limited period of time and to some specific extent. Excessive time pressure leads to stress and negative motivation of personnel coupled with a decrease of work performance to below the standard level.

On the other hand, not enough time pressure results in underperformance of the staff, who may use extra time for extending social and other activities. Boredom due to working requirements that are to low may, also decrease motivation and thus, productivity.

These effects are expressed in Fig. 3, which shows a time pressure factor measured on a non-dimensonal nonnegative scale and depending on the deviation x of planned time t^* to elapsed time t for the current activity, i.e. $x=t-t^*$. If the activity is behind schedule ($t>t^*$), then first positive and later negative effects take place. If the fulfillment of the plan is easy ($t<t^*$), boredom shows some negative effects. Such a relationship can be expressed, superimposing two logistic functions, i.e., functions of the type

$$y_{1,2}(x) = a + b/(1+\exp(c\,(d-x)));$$

$$tp = y_1(t-t^*)+y_2(t-t^*)),$$

one of them expressing the effects of boredom, the other one reflecting the effects of a backlog with respect to the planning. These two functions are calibrated with assumptions on the standard level (tp=1) for a deviation x=0, assumptions on the activation levels for extreme boredom, extreme backlog, and peak performance, the steepness of stepping inside these effects, and the mutual deferment of the effects. In Fig. 3, for instance, the boredom level is approx. 0.75, the overload level is approx. 0.63, the peak level 1.13, the steepness toward boredom level is smaller than towards overload level, and the deferment between these transitions is about 12 on the x-axis (which may be measured in working days). Note that similar functional relationships of transitions between different states can be found for various biological (and other complex) systems such as, for instance, cells.

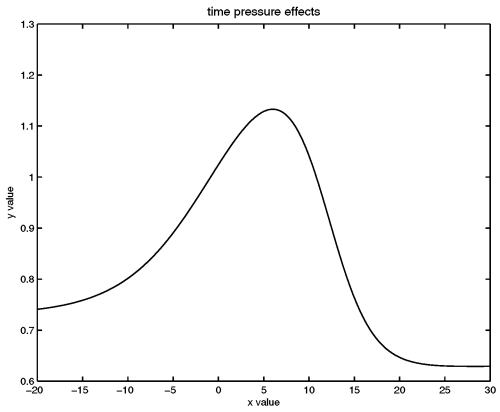


Fig. 3: Plan Deviation and Time Pressure

For a practical application of this model, it is, however, not easy to obtain the values of t and t^* and other parameters for fitting the function of tp. Assuming that there is not a specific set of tasks assigned to a person, individual delays are balanced by team colleagues being faster. Then, time pressure only results if the team's accomplished workload is behind the planned workload. Or in other words: The workload managed at the current time t is larger than a planned or estimated time t^*. Therefore, it is necessary to estimate the time needed for each task prior to the start of the simulation. During a simulation run, it is necessary to do some bookkeeping of accomplished tasks, i.e., an updating of t^*.

Let us note that the basic characteristics of the above function correspond to general assumptions on the psychological relationship between activation and performance according to (Welford 1970).

INSPECTION TEAM PERFORMANCE

While most activities in software development are assumed to be done on a rather individual basis (i.e., one task is assigned to one person), code inspections (Ebenau and Strauss 1994) are usually done by a team of persons working on the same document. Even if inspection in a closer sense (i.e., reading the document) is done individually, the outcome obtained by a meeting where the errors found are collected is group-specific, since the collectively found number of errors is usually not the sum of individually found errors (esp. because of double-counting).

For an individual inspector k, it is assumed that the probability of finding a defect is a result of his or her defect detection skill, $dds_k \in [0,1]$, multiplied by a factor, itf, expressing the effectiveness of the specific inspection technique and the influence of other organizational factors. Empirical results of the influence of such organizational factors on the inspection results are provided, e.g., in (Biffl and Gutjahr 2001).

Applying the individual defect detection probability, inspector k's individual probability of overlooking a defect is $(1 - itf\ dds_k)$. For the inspection team, the effect of double-counting found defects is considered by using the team probability of overlooking a defect (which is the product of the individual probabilities under some independence condition) for calculating the number of found defects:

$$fd^1_i = pd_i(1 - \Pi_{k=1}^{no_inspectori}\ (1 - itf\ dds_k))$$

COMPOUND MODEL AND MODEL FITTING

The most important relationships between human factors, task-(artefact) specific factors, productivity, and outcome-related variables (effort, change of defects) are summarized in Fig. 4.

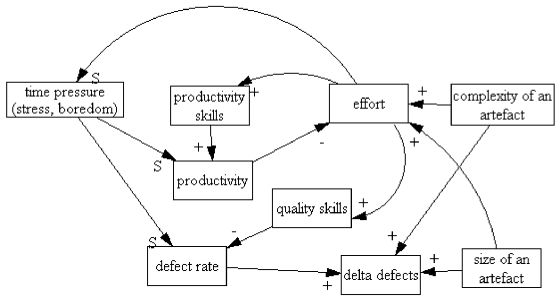


Fig. 4: Rough Influence Model of Personal Factors.

Since the average productivity of programmers can be predicted better than individual ones (Johnson 1998), one may use the concept of a "standard programmer" without extensive collection of data. For instance, assuming a given maximum productivity of a person, mcp, e.g., mcp = 30 lines of code/h, the actual productivity may be given by

$$cp_k = tp_k\ mcp\ cps_k$$

An average productivity may then serve to calculate typical values for cp_k, assuming that the time pressure factor is 1 on average. The organizational factors are expressed in the values for maximum productivities, etc., which are assumed to be fixed for the given project. For setting these values of the given model, real-life data from industrial software development processes should be used.

STOCHASTIC EFFECTS

Even after careful refinement of the quantitative relationships in a model of human effects in software development processes, there are essential impacts on productivity that cannot be fully determined a priori, for instance, individual factors such as fatigue, boredom, and other physical and mental factors. These human factors may be considered by stochastic elements in the process, which influence, for instance, the working times and numbers of defects produced (quality of work). Therefore, the results of an activity with respect to the changed number of defects and the time needed for performing it are considered as being partially random.

For this reason, the above productivity and quality variables (according to a deterministic model) are multiplied by random factors. These factors are assumed to be stochastically determined according to a lognormal distribution with an expected value 1. One main reason for choosing this type of distribution is that typically, undershootings of the expected value are rather small while overshootings may occasionally become tremendous. Other aspects of using the lognormal distribution in modelling are, for instance, discussed in (Mitzenmacher 2001, Jordan 1995).

Multiple runs of the model can be used for estimating the effects of such random influences on the distribution of the model outputs, especially with respect to the objectives quality, project duration, and costs. Using such information, a project manager may get a better feeling for the risks within a scheduled project, e.g., by considering a worst-case scenario within a 95% confidence interval.

CONCLUSIONS

In this paper, various human factors influencing productivity in software development processes are considered. As the main factor, learning and effects related to time pressure are discussed. Many other human factors have been neglected up to now. Their impact is considered by stochastic variables that allow, for instance, to perform a series of simulation experiments. Because of lack of space, numerical results of our complete simulation model of software development processes could not be presented within this paper. Results of a preliminary version of the model can, however, be found in (Neu et al. 2002, 2003).

For the future it is planned to refine the models and validate them with empirical data. As discussed in (Lang et al. 2003), such data should also allow for determination of complex relationships between influencing and dependent variables, e.g., by tools like neural networks. Unfortunately, up to now, practically no appropriate empirical data on human-based variables has been collected in a systematic way in software developing companies. Validated numerical relationships are also a pre-requisite for reliable optimization of software development processes, see, e.g., (Hanne and Nickel 2003).

ACKNOWLEDGEMENTS

This research was supported by the German Bundesministerium für Bildung und Forschung (project SEV) and the Stiftung Rheinland-Pfalz für Innovation (project ProSim, project no. 559).

REFERENCES

Abdel-Hamid, T., and S. E. Madnick, 1991. *Software Project Dynamics - An Integrated Approach.* Prentice-Hall, Englewood Cliffs.

Biffl, S. and W. Gutjahr, 2001. "Influence of Team Size and Defect Detection Technique on Inspection Effectiveness". In *Proc. of IEEE Int. Software Metrics Symposium*, London, April 2001, 63-77.

Christie A.M. and Staley M.J., 2000. "Organization and Social Simulation of a Software Requirements Development Process". *Journal of Software Process Improvement and Practice* 5, 103-110.

Ebenau, R.G.; S.H. Strauss, 1994. "Software Inspection Process" McGraw-Hill, New York.

Hanakawa, N.; S. Morisaki; and K. Matsumoto, 1998. "A learning curve based simulation model for software development". In *International Conference on Software Engineering (ICSE) 1998*, 350-359

Hanakawa, N.; K.-i. Matsumoto; and K. Torii, 2002. "A Knowledge-Based Software Process Simulation Model". *Annals of Software Engineering* 14, No. 1-4, 383-406.

Hanne, T. and S. Nickel, 2003. „A multi-objective evolutionary algorithm for scheduling and inspection planning in software development projects". Report of the Fraunhofer ITWM 42, Kaiserslautern.

Johnson, L. F., 1998. "On Measuring Programmer Team Productivity". In *University and Industry - Partners in Success.* Conference Proceedings, Vol. II, ed. by R. Hornsey and P. Eng. IEEE Service Centre, Piscataway, NJ.

Jordan, M. I., 1995. "Why the logistic function? A tutorial discussion on probabilities and neural Networks". Technical Report.

Kellner, M. I.; R. J. Madachy; and D. M. Raffo, 1999. "Software process simulation modeling: Why? What? How?" *Journal of Systems and Software* 46, No. 2-3, 91-105.

Lang, P.; A. Sarishvili; and A. Wirsen, 2003. "Blocked neural networks for quantification of relationships of software development processes." Report of the Fraunhofer ITWM, Kaiserslautern.

Mitzenmacher, M., 2001. "A Brief History of Generative Models for Power Law and Lognormal Distributions". Working paper, Allerton,

Neu, H., T. Hanne, J. Münch, S. Nickel, A. Wirsen, 2002. "Simulation-based risk reduction for planning inspections." In *Product Focused Software Process Improvement,* M. Oivo and S. Komi-Sirviö (Eds.): Springer (Lecture Notes in Computer Science, 2559), Berlin, 78-93.

Neu, H., T. Hanne, J. Münch, S. Nickel, A. Wirsen, 2003. "Creating a code inspection model for simulation-based decision support." Paper presented at ProSim '03, Portland State University, May 3-4, 2003.

Rus, I., 1998. *Modeling the Impact on Cost and Schedule of Software Quality Engineering Practices*, Ph.D. Dissertation, Arizona State University.

Sherdil, K., N. H. Madhavji, 1996. "Human-oriented improvement in the software process," Working paper.

Sterman J.D., 2000. *Business Dynamics, Systems Thinking and Modelling for a Complex World.* McGraw-Hill.

Welford, A.T., 1970. "On the nature of skill" In *Skills*, D. Legge (Ed.). Harmondsworth, 21-32

OPTIMAL RECRUITMENT IN A MARKOVIAN MANPOWER PLANNING SYSTEM

Süleyman Özekici
Department of Industrial Engineering
Koç University
Sarıyer-İstanbul, Turkey
e-mail: sozekici@ku.edu.tr

İsmail Kocaman
Honda Türkiye A.Ş.
Turgut Özal Cad. No:123
Gebze-Kocaeli, Turkey
e-mail: ismail.kocaman@honda-eu.com

KEYWORDS
Markov chain, manpower planning, random environment, optimal recruitment, linear programming

ABSTRACT

An important decision problem in manpower planning concerns the determination of the number of new recruits who should enter an organization in a given period. This decision must of course be based on various cost factors involved and the manpower requirements of the organization. Another important issue is that the decision maker should also consider the prevailing economic conditions. We propose an LP formulation to minimize the average cost per period in a Markovian model that also incorporates the effect of economic variations.

INTRODUCTION

Managerial strategies of organizations on all functional and operational issues are always affected by the state of the economy. Decision models should therefore take this into account and consider the variations which generally have a random structure. The state of the economy can often be classified in various ways with respect to macro economic indicators. In an expanding economy or market, for example, organizations will tend to grow while the opposite may be true if the economy is in a recession. One of the most important decisions faced in any organization concerns the number of new employees who should be recruited for each job category. This decision must of course be based on various cost factors involved and the manpower requirements of the organization. An equally important issue is that the decision maker should also consider the prevailing economic conditions. All other parameters in manpower planning models involving transfers and wastages depend on the state of the economy. We will use the generic term "environment" to denote the exogenous economic factors that affect the model parameters. The environmental process is then the random sequence of factors of variation that the system is subjected to. This has been the central theme in Özekici and Kocaman (1996) who propose a generalized Markov model in manpower planning where the model parameters are modulated by a Markov chain. They extend the available results in the literature by providing computationally tractable procedures to analyze the transient and stationary behaviour of the system. Technically speaking, these simple procedures usually involve linear algebra. Our aim in this article is to exploit these results and analyze an optimal recruitment problem that has a linear programming (LP) formulation.

Variations on this problem, in one form or another, has been considered by many in the literature. Mehlmann (1980) uses dynamic programming to determine the optimal recruitment and transition strategies. Morgan (1979) considers a model with several job categories and types of recruits to determine the best mix of qualifications amongst the recruits. Algorithms to calculate optimal control policies for various assumptions on the time horizon, staff distribution, transience and stationarity are discussed in Grinold and Stanford (1974). An optimization problem to minimize the present value of the total cost of system operation subject to restrictions on the stocks, inflows and the size of the system is analyzed in Grinold (1976). Tsantas and Vassiliou (1993) and Tsantas (1995) also consider the probabilistic analysis and control of a non-homogeneous Markov system in a stochastic environment.

In Section 2, we present a brief overview of the generalized manpower planning model of Özekici and Kocaman (1996). The optimal recruitment problem is formulated in Section 3 and a numerical illustration is provided in Section 4.

THE STOCHASTIC MODEL

Let Y_t denote the state of the environment (economy) in period t and suppose that $Y = \{Y_t; t = 0, 1, 2, ..., \}$ is a Markov chain with states $1, 2, \cdots, z$ and transition matrix Q. We suppose that Y is ergodic with limiting distribution $\pi(x) = \lim_{t\to+\infty} P[Y_t = x]$ which can be calculated as the unique solution of the system of linear equations $\pi = \pi Q$ with the normalizing condition $\sum_{x=1}^{z} \pi(x) = 1$.

We assume that the organization has k job categories

where each individual moves independently of others. If the environmental state is x at the beginning of a period, then an individual who is in category i at the beginning of the period will be in category j at the next period with probability $P_x(i,j)$ or leave the system never to return with probability $w(x,i) = 1 - \sum_{j=1}^{k} P_x(i,j)$. The number of people who are recruited in environment is denoted by $R(x)$, and each one enters the system in category i with probability $r(x,i)$. Here, P, w and r are called the transfer matrix, wastage probability vector and recruitment probability vector respectively. Note that $R(x)r(x,i)$ is the expected number of new recruits who enter category i in environment x.

Let $N(x,i)$ denote the expected number of individuals in category i when the environment is x under steady state conditions and define N^* as the $k \times z$ row vector

$$N^* = [N(1,1), \cdots, N(1,k); \cdots; N(z,1), \cdots, N(z,k)] \tag{1}$$

and M^* as the $k \times z$ row vector

$$M^* = [M(1,1), \cdots, M(1,k); \cdots; M(z,1), \cdots, M(z,k)] \tag{2}$$

where

$$M(y,j) = \sum_{x=1}^{z} \pi(x) Q(x,y) R(x) r(x,j). \tag{3}$$

If the $(k \times z) \times (k \times z)$ matrix P^* is defined as $P^*[(x,i),(y,j)] = Q(x,y)P_x(i,j)$ such that

$$P^* = \begin{bmatrix} Q(1,1)P_1(1,1) & \cdots & Q(1,z)P_1(1,k) \\ \vdots & \ddots & \vdots \\ Q(1,1)P_1(k,1) & \cdots & Q(1,z)P_1(k,k) \\ \vdots & \ddots & \vdots \\ Q(z,1)P_z(1,1) & \cdots & Q(z,z)P_z(1,k) \\ \vdots & \ddots & \vdots \\ Q(z,1)P_z(k,1) & \cdots & Q(z,z)P_z(k,k) \end{bmatrix} \tag{4}$$

then N^* can be calculated as the unique solution of the system of linear equations

$$N^* = N^* P^* + M^* \tag{5}$$

where the solution is

$$N^* = M^*(I - P^*)^{-1} \tag{6}$$

in compact form. For detailed derivations on these results and accompanying explanations, the reader is referred to Özekici and Kocaman (1996).

Although these results are obtained under the assumption that $r(x,i)$ is the recruitment probability vector, they can easily be adjusted for the case where we define $R(x,i) = R(x)r(x,i)$ as the number of individuals who are recruited for job i in environment x. Therefore, in the new setting where we focus on the optimal recruitment problem, $R = \{R(x,i)\}$ represents the recruitment policy. Given such a policy, or set of decision variables, (3) and (5) state that the relationship between the decision variables $R(x,i)$ and other variables $N(x,i)$ and $M(x,i)$ can be described by a system of linear equations. It now follows that (3) can be rewritten in the form

$$M(y,j) = \sum_{x=1}^{z} \pi(x) Q(x,y) R(x,j) \tag{7}$$

while (5) becomes

$$N(x,i) = \sum_{y=1}^{z} Q(y,x) \sum_{j=1}^{k} P_y(j,i) N(y,j) + M(x,i) \tag{8}$$

in open form for any $1 \le x \le z, 1 \le i \le k$. The fact that the relationship is linear among these variables suggest that an LP formulation can be made in the optimal recruitment problem which we discuss next.

THE OPTIMAL RECRUITMENT PROBLEM

The objective is to find the optimal policy that minimizes the average total cost per period by choosing values for $R(x,i)$. This results in the values for $N(x,i)$ and $M(x,i)$ as given in (7) and (8). The cost structure has four components; recruitment, stock, compensation and transfer. They are represented by the following cost parameters:

- $C_1(x,i)$ = recruitment cost/individual who enters category i if the environmental state is x
- $C_2(x,i)$ = stock cost/individual who works in category i if the environmental state is x
- $C_3(x,i)$ = compensation cost/individual who leaves from category i if the environmental state is x
- $C_4(x,i)$ = transfer cost/individual who enters category i if the environmental state is x.

These cost components are quite self explanatory and they describe a general setup. The recruitment cost is due to new recruits entering the system from outside in a given period. The stock cost includes periodic salaries or wages of individuals who worked in the organizational structure. The compensation cost represents the benefits provided to the individuals who leave the organization. Finally, the transfer cost is a result of additional activities, like education, to place individuals in a new job category.

The decisions must be taken to satisfy certain restrictions. There may be upper and lower bounds on R, N and M for each possible job and environment combination. Our cost structure and the fact that the relationship among the decision and state variables are linear

imply that the problem can be formulated as the following LP problem:

$$\min \sum_{x=1}^{z} \sum_{i=1}^{k} \begin{bmatrix} C_1(x,i)R(x,i) + C_2(x,i)N(x,i) \\ +C_3(x,i)M(x,i) + C_4(x,i)T(x,i) \end{bmatrix} \quad (9)$$

such that

$$M(x,i) - \sum_{y=1}^{z} \pi(y)Q(y,x)R(y,i) = 0 \quad (10)$$

$$N(x,i) - \sum_{y=1}^{z} Q(y,x) \sum_{j=1}^{k} P_y(j,i)N(y,j) - M(x,i) = 0 \quad (11)$$

$$N(x,i) - M(x,i) - T(x,i) = 0 \quad (12)$$

$$R(x,i), N(x,i), M(x,i), T(x,i) \geq \text{Lower bounds} \quad (13)$$

$$R(x,i), N(x,i), M(x,i), T(x,i) \leq \text{Upper bounds} \quad (14)$$

for all $1 \leq x \leq z, 1 \leq i \leq k$.

In this formulation, constraints (10) and (11) simply state the relationships between R, N and M as described by (7) and (8) respectively. We define $T(x,i)$ to be the total number of transfers into category i in environment x as $N(x,i) - M(x,i)$ by constraint (12). This is actually for ease of representation rather than necessity since we can always substitute this difference for $T(x,i)$ in the objective function. Finally, constraints (13) and (14) specify some lower and upper bounds on the parameters. In particular, job descriptions and requirements may dictate lower bounds on N while long term company strategies put upper bounds on R. The LP formulation has at most $4(k \times z)$ decision variables and $7(k \times z)$ constraints. It may be easier to solve the dual formulation. There are very efficient algorithmic procedures to solve such an LP problem without much computational difficulty even for problems with rather large k and z values. Once the optimal values of $R(x,i)$ are found, then the total number of individuals who should be hired in a period is $\sum_{1 \leq i \leq k} R(x,i)$ if the environment is x.

A final word of caution in the LP formulation concerns the actual decision problem it represents. Note that the decisions are not based on the number of individuals who are already employed in the given job categories. Thus, $R(x,i)$ depends only on the observed environment x and the job category i without any reference to the number who are already at category i. This approach may require a more complex dynamic programming formulation. Moreover, all quantities are averages over the infinite time horizon. Our decision problem and its optimal solution is aimed to give the decision maker a good guideline to follow in recruitment. The interpretation of the values must be such that $R(x,i)$ is the number of people who should be recruited into job i in environment x in the long run. Depending on prevailing conditions at the time a decision is taken, there may be deviations from this prescription. But, on average, the recruitment should be as specified by R which results, on average, with size N and M. We believe that the optimal solution may provide a useful guideline to follow. Moreover, the main advantage of the LP formulation is that it is computationally tractable even for large problems.

A NUMERICAL ILLUSTRATION

We now illustrate how the formulation in the previous section can be used to solve a given problem. It is based on hypothetical data with $k = 2$ job categories and $z = 3$ environmental states. The transition matrix of the environmental process Y is

$$Q = \begin{bmatrix} 0.40 & 0.50 & 0.10 \\ 0.30 & 0.45 & 0.25 \\ 0.15 & 0.65 & 0.20 \end{bmatrix}$$

and the transfer matrices for the 3 environments are

$$P_1 = \begin{bmatrix} 0.40 & 0.30 \\ 0.30 & 0.50 \end{bmatrix} \quad P_2 = \begin{bmatrix} 0.20 & 0.70 \\ 0.60 & 0.10 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 0.40 & 0.30 \\ 0.60 & 0.15 \end{bmatrix}$$

while the cost functions are

$$C_1 = \begin{bmatrix} 4 & 5 \\ 5 & 8 \\ 6.5 & 4 \end{bmatrix} \quad C_2 = \begin{bmatrix} 10 & 16 \\ 12 & 19 \\ 13.5 & 10 \end{bmatrix}$$

$$C_3 = \begin{bmatrix} 6 & 8 \\ 9 & 15 \\ 7.5 & 8 \end{bmatrix} \quad C_4 = \begin{bmatrix} 1 & 3 \\ 2.5 & 5 \\ 3 & 2.5 \end{bmatrix}.$$

We suppose that there are lower bounds only for N and M as follows:

$$N_{\text{low}} = \begin{bmatrix} 25 & 10 \\ 34 & 12 \\ 16 & 7 \end{bmatrix} \quad M_{\text{low}} = \begin{bmatrix} 5 & 3 \\ 6 & 4 \\ 3 & 1 \end{bmatrix}.$$

Solving the system of linear equations $\pi = \pi Q$ with the normalizing condition, we obtain the limiting distribution $\pi = [0.300813, 0.504065, 0.195122]$ and use (7) and (8) to make the formulation.

The LP formulation is now obtained by minimizing the objective function

$$\begin{aligned} &4R(1,1) + 5R(1,2) + 5R(2,1) + 8R(2,2) \\ &+6.5R(3,1) + 4R(3,2) + 10N(1,1) + 16N(1,2) \\ &+12N(2,1) + 19N(2,2) + 13.5N(3,1) + 10N(3,2) \\ &+6M(1,1) + 8M(1,2) + 9M(2,1) + 15M(2,2) \\ &+7.5M(3,1) + 8M(3,2) + T(1,1) + 3T(1,2) \\ &+2.5T(2,1) + 5T(2,2) + 3T(3,1) + 2.5T(3,2) \end{aligned}$$

subject to the constraints

$$\begin{aligned}
0.12033R(1,1) + 0.15122R(2,1) \\
+0.02927R(3,1) &= M(1,1) \\
0.12033R(1,2) + 0.15122R(2,2) \\
+0.02927R(3,2) &= M(1,2) \\
0.15041R(1,1) + 0.22683R(2,1) \\
+0.12683R(3,1) &= M(2,1) \\
0.15041R(1,2) + 0.22683R(2,2) \\
+0.12683R(3,2) &= M(2,2) \\
0.03008R(1,1) + 0.12602R(2,1) \\
+0.03902R(3,1) &= M(3,1) \\
0.03008R(1,2) + 0.12602R(2,2) \\
+0.03902R(3,2) &= M(3,2)
\end{aligned}$$

$$\begin{aligned}
0.84N(1,1) - 0.12N(1,2) \\
-0.06N(2,1) - 0.18N(2,2) \\
-0.06N(3,1) - 0.09N(3,2) &= M(1,1) \\
-0.12N(1,1) + 0.80N(1,2) \\
-0.21N(2,1) - 0.03N(2,2) \\
-0.045N(3,1) - 0.0225N(3,2) &= M(1,2) \\
-0.20N(1,1) - 0.15N(1,2) \\
+0.91N(2,1) - 0.27N(2,2) \\
-0.26N(3,1) - 0.39N(3,2) &= M(2,1) \\
-0.15N(1,1) - 0.25N(1,2) \\
-0.315N(2,1) + 0.955N(2,2) \\
-0.195N(3,1) - 0.12N(3,2) &= M(2,2) \\
-0.04N(1,1) - 0.03N(1,2) \\
-0.05N(2,1) - 0.15N(2,2) \\
+0.92N(3,1) - 0.12N(3,2) &= M(3,1) \\
-0.03N(1,1) - 0.05N(1,2) \\
-0.175N(2,1) - 0.025N(2,2) \\
-0.06N(3,1) + 0.97N(3,2) &= M(3,2)
\end{aligned}$$

$$\begin{aligned}
N(1,1) - T(1,1) - M(1,1) &= 0 \\
N(1,2) - T(1,2) - M(1,2) &= 0 \\
N(2,1) - T(2,1) - M(2,1) &= 0 \\
N(2,2) - T(2,2) - M(2,2) &= 0 \\
N(3,1) - T(3,1) - M(3,1) &= 0 \\
N(3,2) - T(3,2) - M(3,2) &= 0
\end{aligned}$$

$$\begin{aligned}
N(1,1) \geq 25 \quad & N(1,2) \geq 10 \\
N(2,1) \geq 34 \quad & N(2,2) \geq 12 \\
N(3,1) \geq 16 \quad & N(3,2) \geq 7 \\
M(1,1) \geq 5 \quad & M(1,2) \geq 3 \\
M(2,1) \geq 6 \quad & M(2,2) \geq 4 \\
M(3,1) \geq 3 \quad & M(3,2) \geq 1
\end{aligned}$$

with the additional nonnegativity conditions for all variables. The formulation has 24 decision variables and 30 constraints. This LP can be solved easily by any one of the available software to obtain the optimal solution given in Table 1.

The total average cost is found to be 3038.75. Note that most of the values are non-integers since they represent averages over the infinite planning horizon.

Table 1. Optimal solution of the example problem

(x,i)	$R(x,i)$	$N(x,i)$	$M(x,i)$	$T(x,i)$
$(1,1)$	29.599	25.000	8.374	16.626
$(1,2)$	21.758	20.622	3.000	17.622
$(2,1)$	31.825	40.777	11.670	29.106
$(2,2)$	2.165	31.458	4.000	27.458
$(3,1)$	0	16.000	4.900	11.099
$(3,2)$	1.864	12.024	1.000	11.024

REFERENCES

[1] Gringold, R.C. and R.E. Stanford. 1974. "Optimal Control of a Graded Manpower System". *Management Science* 20, 1201–1216.

[2] Gringold, R.C. 1976. "Input Policies for a Longitudinal Manpower Flow Model". *Management Science* 22, 570–575.

[3] Mehlman, A. 1980. "An Approach to Optimal Recruitment and Transition Strategies for Manpower Systems Using Dynamic Programming". *Journal of the Operational Research Society* 31, 1009–1015.

[4] Morgan, R.W. 1979. "Some Models for a Hierarchical Manpower System". *Journal of the Operational Research Society* 30, 727–736.

[5] Özekici, S. and İ. Kocaman. 1996. "A General Markov Chain Model on Manpower Planning". Technical Report, Department of Industrial Engineering, Boğaziçi University, İstanbul.

[6] Tsantas, N. 1995. "Stochastic Analysis of a Nonhomogeneous Markov System". *European Journal of Operational Research* 85, 670–685.

[7] Tsantas, N. and P.-C.G. Vassiliou. 1993. "The Nonhomogeneous Markov System in a Stochastic Environment". *Journal of Applied Probability* 30, 285–301.

BIOGRAPHY

SÜLEYMAN ÖZEKİCİ received his PhD degree in Industrial Engineering and Management Science from Northwestern University in 1979. He is a professor of Industrial Engineering at Koç University in İstanbul, Turkey where he teaches courses on operations research, stochastic models and processes. His research focuses mostly on the application of stochastic processes in a number of areas including inventory management, queueing, reliability and risk analysis, and financial engineering.

SIMULATION TOOL TO ANALYSE GENDER DIVERSITY AT WORKPLACE

Ignacio J. Benítez
José L. Díez
Juan A. Lacort
Pedro Albertos
Department ISA
Polytechnic University of Valencia
Camino de Vera, s/n 46022 Valencia, Spain
E-mail: pedro@isa.upv.es

Carlos Candela
Institut Universitari d'Estudis de la Dona
University of Valencia
Blasco Ibáñez, 32
46010 Valencia, Spain

KEYWORDS
Simulation Tool, Gender Diversity, Decision Support Systems, Social Science, Gender Barriers.

ABSTRACT

The purpose and operation of the software simulation tool "Analysis of Gender Diversity at a Work Team (Divers@T)" is described in this paper. It has been designed and implemented as a major objective of the project *Divers@: Gender and Diversity*, included within the European Union EQUAL initiative and partly funded by the European Social Fund (Project ES296). This software tool builds the conditions prior to a selection process, according to the information provided by the user, from a gender equality point of view. The needed information focuses on three areas: characteristics of the job, current situation of the labour market or offer, and preferences of the people that make the selection, assigning more or less importance to each of the criteria taken into account when selecting applicants. The users can analyse the resulting scenario, receiving helpful information that could assist them with the selection. A diagnosis of the situation of the institution according to gender equal opportunities is included, and also the option to make a selection and receive a degree of adequacy according to the current environment. Actions to solve a situation of discrimination are presented to the user, so an improvement in gender equal opportunities, when needed, can be achieved.

INTRODUCTION

Traditionally, women have been excluded from executive positions within an institution due to some reasons, being mainly gender discrimination behind most of them. Women encounter quite often difficulties when accessing or promoting to upper positions. Also, women tend to find themselves excluded from the executive actions and the communication networks, traditionally in hands of men (Metcalfe and West, 1995; Simpson, 2000).

The project called *Divers@: gender and diversity* aims to find the origins of these problems of discrimination and help them to disappear. The aim of this project, being currently developed, is to achieve or improve gender diversity at work teams, especially at executive positions, by increasing the presence of women and the detection and eradication of specific so-called "gender barriers", by improving the working conditions and applying the equal opportunities principle.

In order to assist with this main objective, a software simulation tool has been implemented. This simulation tool is mainly focused on the reasons why in most cases women are not fairly represented, and on the selection criteria lying behind the decisions made when promoting or hiring workers.

The criteria so far studied are the diversity principle, according to a previous study by the experts of the characteristics of the position offered, and the diversity of the 'labour market', that is, the proportion of women and men that are eligible applicants. The last is based on: an analysis of the offer, the level of expertise and capabilities of the applicants, and the barriers that workers (especially women) may find in their way to promotion or recruitment.

This simulation tool, which is presented in the next sections, has been implemented using different development environments. For the front-end of the application, Visual Basic has been used, allowing to easily provide all the features expected in a well-designed Graphic User's Interface (GUI). The application guides the user through the different sections in a sensible way, requiring the compulsory information and suggesting default values when applicable, while ensuring always that the entered data is in the appropriate format and values are within the valid range.

Intermediate results are shown to allow the user either to verify, amend or tune the entered data after comparing the results obtained on the simulation, with the ones known from the real world.

Subsequent simulations can be done, starting from the previously entered information, by modifying values, since the application always allows the user to go back and edit the previously entered information, and thus different scenarios can be analysed and their outcomes evaluated. The results obtained in the simulation are compiled in a report, that is saved on a text document on user's demand, and that includes all the information the experts found relevant for the user.

The calculation is made in the back-end, taking advantage of the power of Matlab™. Functions stored in M files

provide output results from the input parameters and in many cases print a graph that will be displayed embedded as part of the GUI, or illustrating the written report.

A COM component, generated with the Matlab COM Builder, is used to communicate the simulation tool and the Matlab M files, passing the information entered by the user in the front-end to the back-end, where the appropriate calculation will be made, and sending the results back to the front-end for user's evaluation.

Following, there is the explanation of the software tool and its possibilities, an example with a given work team, a conclusions section and a list of references.

SOFTWARE DESCRIPTION

Objectives

The purpose of this tool, as mentioned before, is to simulate the process of creating a work team from the perspective of a user who is assumed to be a Human Resources Department responsible, a person in charge, a researcher trying to measure and obtain conclusions, or simply any person with the intention to use the software, taking into account some statistical measures and also subjective decisions that apply when choosing the workers. The more accurate the input information is, the more realistic the results and analysis will be.

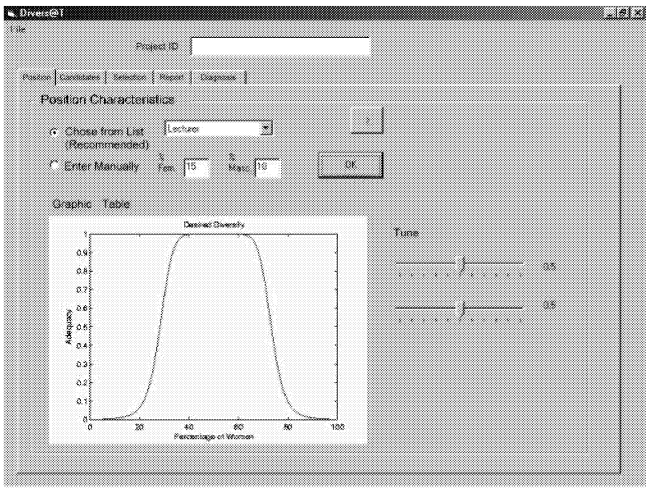
The program starts with a study of the desirable proportion of women and men at the team to achieve gender diversity. A range of percentages is presented, according to the study of the characteristics and needs for the job, and the principles of equality and diversity. A screenshot of the software tool can be seen in Figure 1.

Next, an analysis of the offer is done, taking into account the initial offer and the proportion of men and women, an optional classification according to level of studies, expertise or knowledge, and an optional limitation based on barriers. The experts estimate the grades of barriers by inquiring of workers through specific questionnaires and interviews, trying to capture the extent or absence of obstacles for women at institutional, cultural, organizational and personal areas.

Next is the selection process, where the user can assign different weights to different criteria used in the selection. The software tool takes into account two criteria. These are the selection of women and men according to the diversity principle, and the selection of men and women according to the analysis of the offer, levels of expertise and barriers.

Finally, the program shows the results of the analysis and suggests optimal diversity percentages, according to the availability of women applicants under the current circumstances. At this point, the user has the option to make a selection of women and men for the team. Then, the program will show the user a measure of adequacy between the selection done and the results of the simulation.

A final diagnosis section is also included, which evaluates the situation of the offer and the barriers encountered. Steps are being taken, so the program will suggest specific actions that should be followed in order to achieve a better diversity and equal conditions, as well as an improvement in productivity and client-based relations.



Figures 1: Positions Tab of the Simulation Tool

The methods to obtain the two functions of diversity for the selection criteria are described next.

Desired Diversity

There would be different ways to implement diversity (Benítez et al. 2003, and the references therein). The one chosen here is by means of a modified generalized bell function, such as the one in Figure 2. The generalized bell function has the formula shown in Equation (1).

$$y = \frac{1}{1 + \left| \frac{x - c}{a} \right|^{b}} \qquad (1)$$

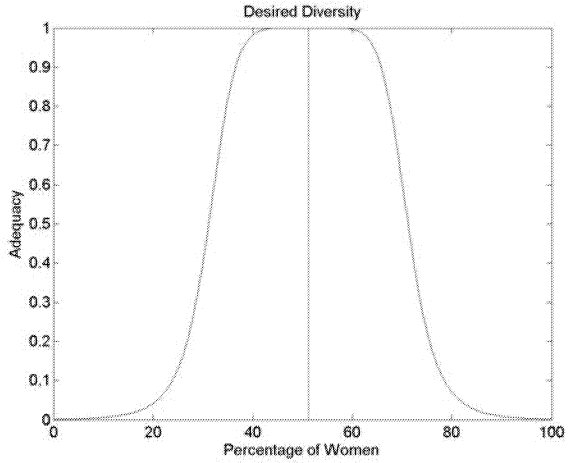
It has two parameters. Parameter c denotes the centre of the bell and parameter a determines the width of the function. By b, which is a function of a and c, the bell is shifted into the available range (0 to 100%), fixing the slope of the diversity function.

There exists a great amount of documentation about the effectiveness of using generalized bell functions for membership functions in fuzzy logic, and how these three parameters can be used to depict the bell function according to the specifications (Jang et al. 1997; Abraham and Nath 2000). In the tool, both parameters a and b can be slightly modified to fine tune the width and slope of the resulting bell (the maximum modification allowed being +/- 10 % of their original values).

The Desired Diversity function yields an optimum range of Women/Men proportion at the work team, according to the principles of equity and diversity, and a sterotyped classification of the characteristics needed for the job. This classification is made by the experts into three categories: "Masculine", "Feminine" and "Neutral" characteristics, and takes into account information such as the tasks definition for the specific job, the gender of the clients and the importance of some skills or abilities compared to others.

The results are percentages of "Masculine", "Feminine" and "Neutral" characteristics. This classification is based on documentation that states the existence of generic differences in abilities and skills for women and men (Barberá et al. 2000; Kite 2001).

A collection of characteristics is provided with the tool. The user can either use them, modify them, create new definitions or assign weights to characteristics according to the importance they may have when profiling a position.



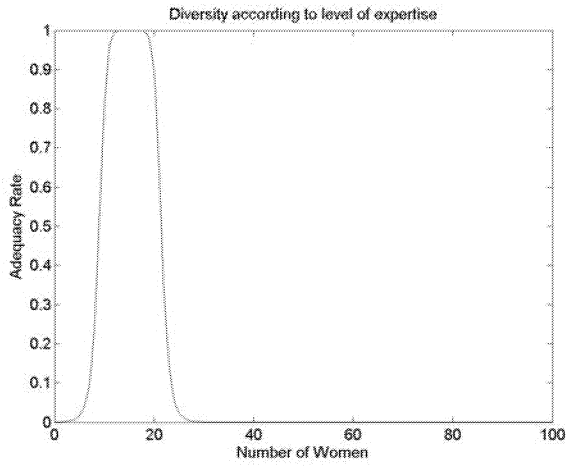
Figures 2: Example of Desired Diversity

For instance, in Figure 2, a previous study of the position states that its characteristics are 22 % "Feminine", 14 % "Masculine" and 64 % "Neutral". Assigning 0.7 and 0.3 to the two extra parameters for the slight variation in width and slope (both of them can vary between 0 and 1, being 0.5 the default value), the result is a bell function whose centre is at 51.20 % of women, and a validity scope of +/- 10 % (See Figure 2). This means that, according to desired diversity, the proportion of women at the work team will be optimal at any value between 41.20 and 61.20 % of women (the range where the bell function presents its maximum).

Diversity According To Offer And Level Of Expertise

Another criterion used in the selection is the analysis of the offer. The result of this analysis is a range of percentages of women suitable to be hired or promoted to the work team, according to the gender proportion of the initial set of candidates, the availability of applicants according to levels of expertise or knowledge, and the presence of barriers (cultural, organizational, personal) limiting the access. This diversity function could be like the one shown in Figure 3, and is again a modified generalized bell function, with two extra parameters to slightly vary width and slope by the user. It is obtained after a step by step process, starting with the initial offer.

Offer.
An estimation of the maximum number of applicants meeting minimum requirements is needed. Introducing, as well, the proportion of men and women in this group of applicants will establish how balanced/unbalanced the offer is according to gender.



Figures 3: Example of Diversity based on Offer Analysis

Expertise, Professional Background.
Once the offer has been typed, different levels of expertise, knowledge or academic background of the candidates may be introduced, in order to cluster or differentiate among applicants, how many of them meet not only minimum requirements, but more qualifications that would be not essential but highly desirable. The values are entered as a matrix of two columns, one for women and one for men, and as many rows as levels are stated, starting from row 1, which would be the level with minimum requirements. Percentages of women and men fulfilling expectations for each level would be introduced. It is expected that the higher the level is, the smaller the percentage will be for both women and men. An example can be seen in Table 1.

Table 1: Classification According to Levels of Expertise

Level	Women (%)	Women (#)	Men (%)	Men (#)
1	100	338	100	789
2	62	210	75	592
3	24	81	44	347
4	2	7	6	47

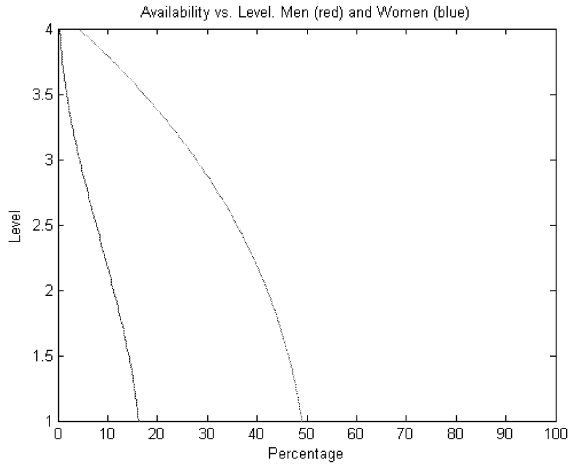
Barriers.
After levels have been introduced, the user has the option to include a limitation based on barriers. Different indices of barriers can be assigned according to gender and level. These indices are introduced as coefficients between 0 (barriers do not exist) and 1 (barriers totally block access), and are obtained after a meticulous study of the situation of the institution from a gender equality point of view. This study is based on information gathered by the experts, and will be included as part of the software tool, though this work is still in progress.

As a result of the presence of barriers, percentages of men and women available at each level can be reduced. The resulting percentages are obtained as a product of the percentage by the complement of the barrier index. An example can be seen in Table 2.

The results are then used to generate two availability functions constrained after barriers, as can be seen in Figure 4, where the function on the right is the availability for men, and the one on the left is the availability for women.

Table 2: Effect of Barriers Indices on Availability

Level	Candidates (%)		Barrier Index (0..1)		Result (%)	
	Women	Men	Women	Men	Women	Men
1	100	100	0,46	0,3	54	70
2	62	75	0,4	0,2	37,2	60
3	24	44	0,4	0,1	14,4	39,6
4	2	6	0,3	0	1,4	6



Figures 4: Availability Functions of Men and Women according to Level

These functions are polynomial functions whose order is the number of defined levels. The parameters are obtained using the Least Mean Squares (LMS) algorithm, whose well-known formula is shown in Equation (2).

$$M = (X^t X)^{-1} X^t Y \qquad (2)$$

Where $M=[m_1\ m_2\ \ldots\ m_n]$ stands for the vector of parameters, and X and Y are matrices of measured input and output data, respectively. The structure of the resulting polynomial functions is shown in Equation (3)

$$y = m_1 x^{n-1} + m_2 x^{n-2} + \ldots + m_{n-1} x^1 + m_n x^0 \qquad (3)$$

where y stands for "percentage" and x stands for "level", being n the order of the polynomial (number of levels).

Requested Level.
If a classification according to levels of knowledge or expertise has been introduced, there exists the option to choose the level of required knowledge. The user can assign a numerical value of desired level, not only an integer, but also a decimal number, even when this could not seem to be a right value. If no specific level is introduced, the most skilled people are selected.

According to the requested level and the functions of availability previously obtained (Figure 4), a new diversity function is generated, which will indicate the optimal available range of women percentages based on the proportion of men and women at the requested level. This is the second diversity function used as another criterion for the selection. An example of this function, depicted in Figure 3, has its centre at 15.21 % of women, and a validity range of +/- 2.5 %

Selection Process

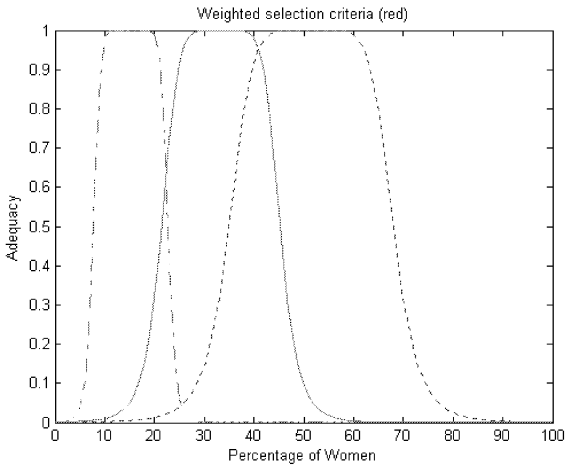
The two diversity functions would be combined at this point to obtain a third one, from the weighted average of the parameters of the two diversity functions (centres and widths), as can be seen in Equations (4), (5) and (6).

$$c_t = w_1 c_1 + w_2 c_2 \qquad (4)$$

$$a_t = w_1 a_1 + w_2 a_2 \qquad (5)$$

$$w_1 + w_2 = 1 \qquad (6)$$

The weights (w_1 and w_2) are introduced by the user, allowing to give more or less weight to one criterion or the other, as a personal user's decision. An example of the weighted selection criteria is depicted in Figure 5.



Figures 5: Example of Weighted Selection Criteria

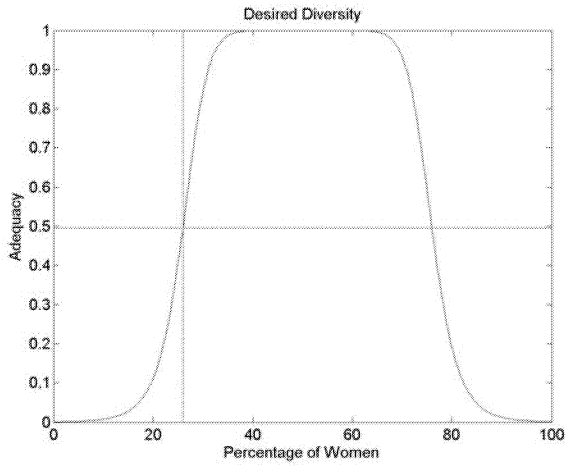
In Figure 5, the function on the left (dash-dot) would be the diversity obtained from the analysis of the offer and barriers, the function on the right (dotted) would be the desired diversity, and the function between the previous ones would be the result of the weighted combination of the two criteria, in this case having the two of them the same weight. This resulting function has the centre at 33 % of women and a validity range of +/- 5 %.

The simulation yields three diversity functions with their parameters. At this point, the user can evaluate the result and obtain conclusions from the analysis. Further steps in order to improve the current situation will depend on user's or institution will and commitment.

Results

Once the user makes a selection, the projection of the number of women selected on the three diversity functions will be displayed, so the user can assess the rightness of the selection done attending to the criteria.

An example is depicted in Figure 6. The adequacy of the selection on a desired diversity function is shown. In this example, for a selection of 26 % of women at the team, the resulting adequacy is 49.61 %, which is half way between the worst (0) and the best possible (1) adequacy.



Figures 6: Example of Adequacy of a Selection on the Diversity Function

EXAMPLE

As an example, let be considered a team with 13 positions. A study of the position yields a percentage of 22 % "Feminine", 14 % "Masculine" and 64 % "Neutral" characteristics. The first stage of the software tool yields the desired diversity function, which can be seen in Figure 2.

The maximum number of possible candidates meeting minimum requirements is 1127 applicants, 30 % of them being women and 70 % men. Information about four levels of expertise is available, so this classification is introduced, and can be seen in Table 1.

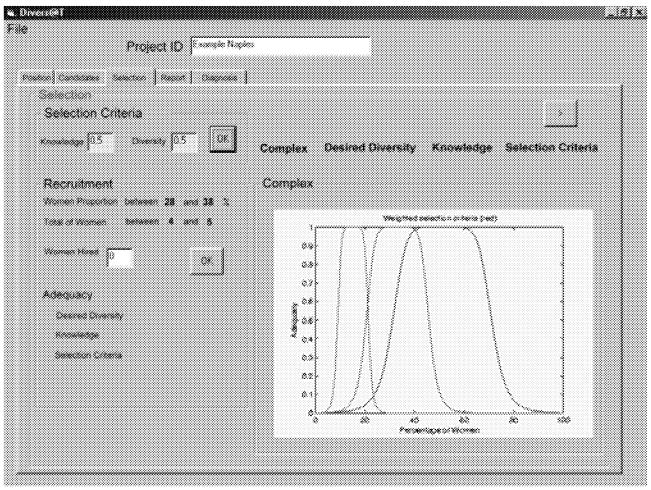
The barriers indices are also available, so they are as well introduced in the simulation tool. The values and the resulting availability functions can be seen in Table 2 and Figure 4 respectively.

A level is chosen for the selection. The user wants to see the results when introducing a desired level of 2.8. The proportions of men and women at the requested level considering barriers is generated, being used to set the parameters of the diversity function according to offer. The values are 15.21 % of all women and 84.79 % of all men applicants. This is an important constraint for diversity.

Changes in offer due to barriers are calculated and shown. This is a valuable information, since it will give us a picture of the scenario and will highlight areas where positive actions would be urgently needed.

Both criteria for the selection are assigned the same weight (0.5 for both of them). The resulting diversity function is depicted in Figure 5. A screenshot of the simulation tool is shown in Figure 7.

Comprehensive information is available. The software tool provides the range of proportion of women for an optimal hiring or promotion according to the resulting diversity function. In this example, this range turns to be 28 to 38 % of women, which translated into positions is 4 to 5 women out of 13 positions. Other information would be the range of proportion to accomplish desired diversity, in this case being between 41.20 and 61.20 % of women (5 to 8 women). The gap between desired and resulting ranges is, in this particular case, a consequence of the poor availability of women at the requested level.



Figures 7: Simulation Tool Results Window

CONCLUSION

A simulation tool has been implemented to assist users willing to analyse positions, offer and barriers of an institution from the point of view of gender diversity. As a result, values for a selection are given and areas suitable for improvement are detected, evaluating the final decision of the user. Based on these results, corrective actions can be suggested by experts to overcome discrimination.

A Graphic User's Interface (GUI) layer has been added, so the user can interact with the software in the easiest possible way.

REFERENCES

Abraham, A. and Nath, B. 2000. "Evolutionary Design of Neuro-Fuzzy Systems – A Generic Framework". In *Proceedings of the 4th Japan-Australia Joint Workshop on Intelligent and Evolutionary Systems*. National Defence Academy. Akira Namatame et al. (Eds.). Japan. ISBN: 07317 05041, 106-113.

Barberá, E., Sarrió, M. and Ramos, A. 2000. "Mujeres y estilos de dirección: El valor de la diversidad". *Intervención Psicosocial*, 9(1), 49-62. (In Spanish)

Benítez, I. J., Albertos, P., Barberá, E., Díez, J. L. and Sarrió, M. 2003. "Equal Opportunities Analysis in the University: The Gender Perspective". In *Eurocast 2003. Cast and Tools for Complexity in Biological, Physical and Engineering Systems*. ISBN: 84-688-0820-2. REGISTER: 341403, 96-98.

Jang, J.S.R.; Sun, C.T. and Mizutani E. 1997. *Neuro Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice-Hall International Inc.

Kite, M. 2001. "Gender stereotypes. An Encyclopedy of women and gender. Sex similarities and differences, and the impact of society on gender". Academic Press , Volume 1 (A-K), 561-570.

Metcalfe, B. and West, M. 1995. "Women managers". In *Women at work*, J. Firth Cozens and M. West (Eds.). Open University Press, Buckingham.

Simpson, R. 2000. "Gender mix and organisational fit: how gender imbalance at different levels of the organisation impacts on women managers." *Women in Management Review*, Vol 15, Nº 1, 5-19.

OBJECT ORIENTATION AND RE-USE

UML BASED FMECA IN RISK ANALYSIS

J. GUIOCHET
GRIMM-ISYCOM/LESIA
CS Eng. Dpt., University Toulouse III
31100 Toulouse, France
jeremie.guiochet@insa-tlse.fr

C. BARON
LESIA
INSA, 135 Av. de Rangueil
31077 Toulouse cedex4, France
claude.baron@insa-tlse.fr

Abstract: Today, as systems become more and more complex, safety is becoming critical. Reducing the risk to an acceptable level with a complete risk management activity is necessary. This paper more precisely focuses on risk analysis; its demonstrate how the use of a risk analysis technic such as the Failure Modes, Effects and Criticality Analysis (FMECA) can be coupled to a object oriented system modeling process in order to guide the designer to exhaustively consider all potential risk, to increase the system security . For the system model, we chose the UML notation, which is now a standard in system and software engineering.

Keywords: Risk analysis, safety, system modeling, object oriented principles, UML

1. Introduction

Today, as systems become more and more complex, safety is becoming critical. Safety, sometimes defined as an absolute property, can now be defined as the property of a system to be "free from unacceptable risk" [13]. Therefore it is necessary to reduce the risk to an acceptable level with a complete risk management activity as presented on figure 1. This approach has been used into different domains; and for example, some of its concepts can be found in the medical norm [12]. Inside the general risk management activity, our studies more precisely focus on risk analysis. The justification of the use of the Preliminary Hazard Analysis (PHA), Failure Modes, Effects and Criticality Analysis (FMECA) and Fault Tree Analysis (FTA) techniques is exposed in [8]. However, the reader can find developed the other aspects of risk analysis in [10, 9]. Those activities are based on a system model. Ideally, the system definition is modelled formally, but the use of formal methods in industrial development of safe systems is still rare. A significant barrier is that many formal languages and formal analysis techniques are unfamiliar and difficult to understand and to apply for engineers. Designers must also communicate between specialists of different domains who usually have their own language. For these reasons, existing techniques must be considered. UML (Unified Modeling Language) notation fulfills these claims, and is now a standard in system and software engineering.

In this paper we will focus on the use of FMECA based on UML models in the risk analysis activity. The first section of this paper positions our works in the research communities on UML and risk analysis. Section two and three expose our method based on UML message exchanges and present generic model of error after an analysis of message failure modes. The two following sections propose a generic FMECA array for a system analysis and validate this on an application to the analysis of messages send by actors such as external devices.

2. Related works

The risk analysis activity recently appeared in the domain of computer systems safety. What one can found as a reference is the terminology about fault forecasting in [15]. It consists in anticipating faults using techniques to identify faults and evaluate their effects. This is close to risk analysis concept and particularly to hazard identification and risk estimation activities. Thus, we focus on fault forecasting studies. Several research works can be found on the notion of component, considering that all components have inputs and outputs (by analogy with electronics components). [24] identify software critical components evaluating the risk factor of each component considering its complexity and the seriousness of its potential failure. [23] also use component models, but use the HAZOP (Hazard Operability) analysis technique to automatically generate fault trees. These aspects meet the SIL (Safety Integrity Level) notion, used in norms [11], but cannot be applied to a system modeled with UML. Indeed, the notion of object is too far from the notion of component [5] to efficiently adopt a similar approach.

Nearer to the object paradigm and to UML, [6] define "critical attributes" of the system objects and study the effects of potential harmful variation of these attributes. This leads first to the notion of critical sub-systems (which can be components) and sec-

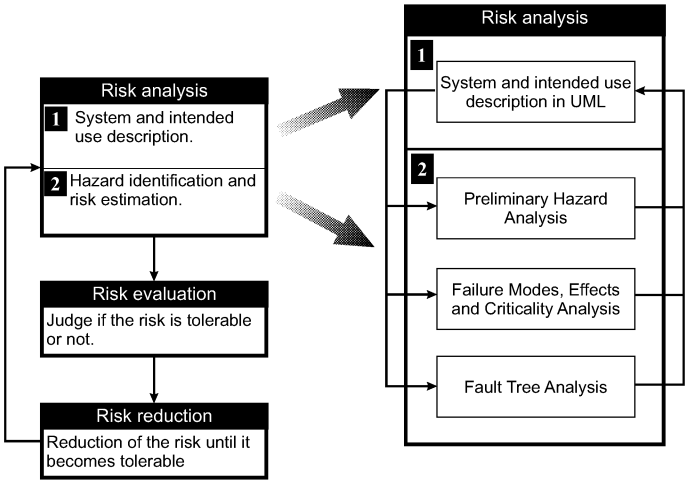


Figure 1 : UML based risk analysis in the risk management activity

ond to the identification of hazardous state based on statecharts ([18]). The approach is the same as the previous ones and consists in identifying parts of the system that might cause some damage. But the link between the very system objects are even more difficult to evaluate.

Among analytical methods, allowing fault forecasting, FMECA ([17]) is certainly the most used during functional analysis. Nevertheless, it can be applied to software components and to their links [24]. This approach is similar to the study of electronics components but does not take into account the major object concepts as classes or methods. [1] suggests to use this technique analyzing the objects methods as a function analysis, and thus identifying effects on the system. In a case study on a car design,[14] use the UML use cases to specify requirements and realize a FMECA based on these diagrams. However, the link with UML is limited because the use cases identified, as the "car stability during braking", correspond to non-functional requirements and thus can not be used to identified objects.

Among analysis techniques, fault tree analysis must also be mentioned. This technique is usually coupled with a FMECA. [7] show how they use fault trees to express the different classes of faults of a model. The connection with the system objects is still complex, and the tree study seems to be done in parallel without any real interaction with the UML models. [22] presents a method and algorithms allowing to automatically generate fault trees from UML models. The resulting trees are in fact reliability models to describe how a failure can occur in parallel to UML models. [2] also elaborated reliability models (*Intermediate Model*) based on UML diagrams, then derived into *Timed Petri Nets* [4, 16, 3]. All these techniques offer tools to designer and overcome some UML weaknesses, as the lack of executing models. However, the techniques mentioned above strongly rely on the notion of component, which is different from the notion of object or class. On this very subject, this paper shows how our approach distinguishes itself from these works, overall because it takes into account different concepts than the component concept.

3. Message failure mode analysis

The notion of failure mode is close to the notion of error; both concepts will be indifferently used in this section.

The FMECA technique consists above all in identifying errors that could occur in a system before its production. Actually, errors are often specific to the application. However, to realize a more systematic error identification step, one can sometimes use some generic error models, which can be applied independently from the application. Unfortunately, these models often concern a few low level elements. Thus, we propose to focus on one of the language element which is not specific to an application: the modeling language itself, and in our approach we chose UML. Indeed, by analogy with electronics devices such as actuators and sensors, the modeling language constructions can be reused from an application to another. Moreover, this multidisciplinary language allows to model either electronic, computing, mechanical elements and even human actors. This genericity is thus double: it should allow to develop generic error models not only for different applications, but also for different domains (from electronics to human components).

Due to UML complexity, we focus on one construction of this language: the notion of *Message*.

Different reasons lead us to this choice. The goal of a failure analysis is to identify hazards during systems use. It is a failure during an activity which lead to a hazard. And the activity of a model depends on the messages. Hazards occur when messages are exchanged. In conclusion, we proceed to a failure analysis based on messages and not on components or on functions.

However, focusing on the notion of *Message* can reduce the genericity. Indeed, the UML notation includes the concept of *Action* which is the fundamental element of the meta-model. Yet, in order to keep our approach close to UML user usual concepts, we preferred basing our approach on the *Message* notion. Moreover, current works on UML formalization show important differences on the *Action* concept. Indeed, version 1.5 of UML [21], strongly modified its previous use (version 1.4 [19]). Furthermore, this specification differs from current works for version 2.0 [20]! On the other hand, the notion of *Message* seems stable in those three documents; it just prove the force but also the importance of this concept.

4. Message error models

This section presents the range of errors linked to the concept of *Message* that we have determined.

Some languages (such as Ada for example) contain an *operational semantics* and a *verification semantics*. The operational semantics allows to specify a system functional aspects and describe how the system will deliver the service. This corresponds to the whole set of UML diagrams. The verification semantics defines properties to verify if some rules are respected. For example, in Ada, constraints on data variables of a software operative part can be expressed. This semantics allows to identify errors resulting from a non-respect of one of these properties, and also permits to treat them. The possible process is thus to group all properties issued from the verification semantics and to derive error models from them.

This approach can be applied to most of languages, but usually, the verification semantics is either implicit, mixed into the operational semantics, either missing. In UML, a certain number of elements can be classified into the verification semantics. First, the use of constraints, graphically represented with curly brackets, allows to specify a restriction on the use of certain kind of elements; an error is raised if the constraint is not respected. There is also in the UML specification the *Well-Formedness Rules* which define a set of constraints expressed with the OCL language [19]. However, most of verification properties are not explicit; they are integrated into the operational semantics. To sum up and conclude on this point, instead of trying to group the set of constraints or *Well-Formedness Rules* together relatively to the notion of *Message*, we propose to generally define the concepts inherent to the notion of *Message*. This approach synthesizes the elements specified in version 1.4 and allows to integrate elements such as time constraints, missing from the meta-model at the moment(work in progress by the OMG for UML version 2.0).

A message can be a signal creation, an operation call, a creation or destruction of an instance; we thus present a generic description for all the characteristics of a *Message*. The graphical representation by a sequence diagram is illustrated on figure 2. The different elements of a message thus are here after defined:

1. the interaction it belongs ;
2. the next and previous messages in the interaction ;
3. the objects that send and receive the message ;
4. the sending and receiving events ;
5. the parameters (number, type and value) ;
6. the implicit response (defined by its arguments, sending and receiving events) ;
7. the period of the message treatment.

Then possible errors for a message are established based on all these elements. First, a message belongs to an interaction, and a sending of a message non-planned is a type of usual error, which often happens in Human-Machine Interface manipulation. Generally, this type of error can be extended with a first error model:

> E.1. *Sending of a message not belonging to the planned interaction.*

The second point dealing with the message order can also lead to errors, particularly with human actors. Indeed, a user having many messages to send might inverse or forget one of them. This type of error can be extended to any model specifying two types of error:

> E.2. *Execution of one or several messages in a wrong order.*

> E.3. *Omission of a message among an interaction.*

A message is sent by an object, but the object supposed to receive it might not exit. This type of error, usual in computer sciences, allows to formulate a generic error:

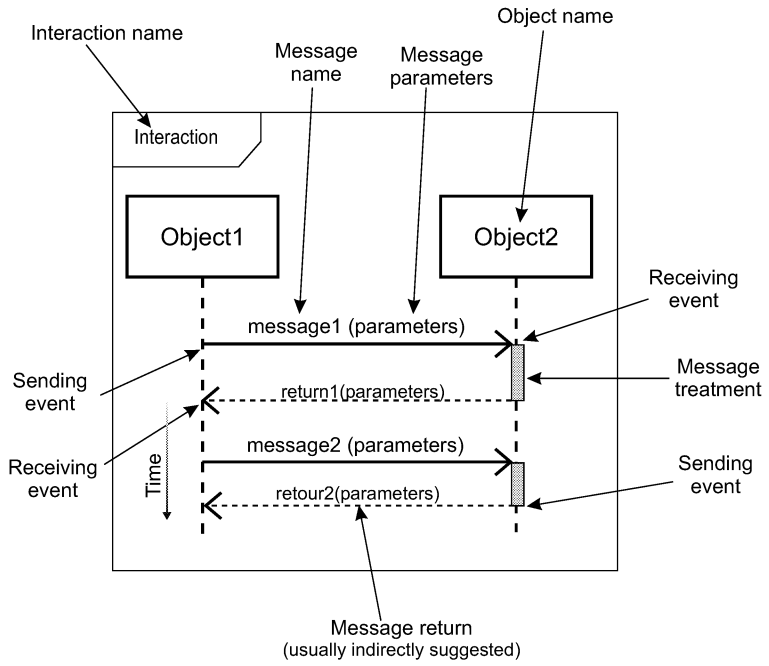


Figure 2 : Elements of an interaction realized by the exchange of two messages

E.4. *Lack of an instance to receive the message.*

Characteristics related to sending and receiving events allow to define temporal properties. Indeed, for these events, time is the fundamental element, and errors are caused by delays; messages can also be in advance compared to their specifications:

E.5. *Sending or receiving of a message outside its specified time limits (too son or too late)*

The message arguments constituting the operation or called signal parameters must correspond (number, type and value) to those expected for the object receiver. This property, partially expressed in OCL by the *Well-Formedness Rules* of the UML specification, allows the expression of tree types of errors:

E.6. *The arguments type is different from the type of parameters expected by the receiver.*

E.7. *The number of a message arguments is different from the number of parameters expected by the receiver.*

E.8. *The value of a message arguments is different from the value of parameters expected by the receiver.*

The usual implicit answer to a message might be characterized by arguments (for example, an message which is an operation call, can return a value) but also by some sending and receiving parameters. This leads to the identification of an error that is generally relative to a message that call an operation (as for example *ReadPosition* for a position sensor):

E.9. *The values returned by a response to a message does not fit with the expected values (for example: constant, random, out of limits, etc.)*

The time of a message treatment correspond to the duration between the receiving of a message and the sending of a response. This response can either be a returned value, or the object contruction/destruction, a signal emission, etc. This type of error can thus be identified:

E.10. *Treatment of a message out of the specified time limits*

Last but not least, in order to complete the elements that are mentioned on the diagram, we must also consider the *link* element, that characterizes the relation between the transmitter and receiver objects, and allows the message emission. This error type can thus be formulate:

E.11. *Lack of link between sender and receiver objects*

5. Proposition of a generic FMECA array for a system analysis

Based on the works and norms for FMECA as [17] (devoted to functional analyzes), this section proposes to introduce the following elements into the FMECA array (cf. figure 3) for a message failure mode analysis:

- the message name,
- the failure modes, or the errors identified thanks to the previous error models,
- the causes of those failure modes,
- the effects at a local level, at a directly higher level and at the system level,
- the data to estimate the risk (*severity* is the damage seriousness, failure mode occurrence noted as *probability*, associated risk),
- the on-line means to detect failure modes and their effects,
- the possible means of risk prevention and protection,
- other pieces of information.

Before explaining the approach, note that the goal in these arrays is not to proceed to a deep analysis of each of the mentioned points; in particular, it is not the point to consider the causes of the causes but to synthesize the main data in order to obtain a system analysis.

The column *Potential solutions* of the array on figure 3 deals with the possible means to reduce the risk. It is important to notice that these means are not directly implemented but this help to focus on the point that a preliminary risk evaluation must be done. Risk is here calculated from a qualitative estimation of the probability of occurrence of a failure mode and of the seriousness of the induced damage. We chose to represent the prevention and protection means but such an analysis can lead to other means. For example, having identified critical messages but for which estimating the probability of occurrence was difficult, a means to reduce the risk is to use fault removal techniques (verification, validation, tests, etc.). More generally, the use of FMECA we propose does not follow a systematic process, where each failure mode is evaluated in terms of probability and seriousness, then treated. The main reason comes from the impossibility to estimate the probability of all failure modes, even in a quantitative way, as in the case of a software analysis. Thus, FMECA is essentially useful to focus on critical and weak design points from the safety point of view.

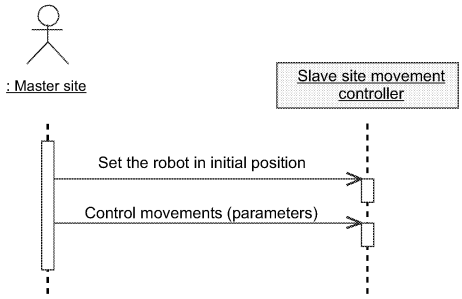


Figure 4 : Sequence diagram illustrating interactions between a master site and a slave site in a telemedicine system

Finally, as FMECA directly depends on the model level of details, its use depends on the development process step applied. In our approach, we recommend to concentrate on the first steps, when safety requirements, architecture choices and major hazards are identified.

6. Application to the analysis of messages send by actors such as external devices

A system approach must allow to take into account all components (electronics, informatics, mechanics and human factors) but it is obvious that only each field specialists really control data. However, as each one has his own language and his own techniques, it can be complex to group information in order to proceed to a global analysis of a system. Using the UML language jointly to the message error models, permits to perform a risk analysis more homogeneous. This section presents an example of use of error models previously identified in order to demonstrate the feasibility of such an approach. This approach has been successfully applied to a complex medical robot system [8].

Objects and interactions modeling An *actor* characterizes an outside user or related set of users who interact with the system [5]. It is possible for an *actor* to be a human user or an external subsystem. This section present an actor of the external device type. In the case of many systems, it is possible to represent external devices as actors if they interact with the system in an autonomous way. For instance, in the case of tele-medicine using robots, where a master site and a slave site interact, the slave site model might include the *master site* as an actor. The messages exchanged with the system can be modeled thanks to a collaboration or a sequence diagram as on figure 4.

Date Author :			Project name						Page : /
Code	Message	Failure mode (error)	Cause	Effects a. Same level b. Upper level c. System level	Risk Severity	Probability	Risk	Possible detection means: (on line) a. Failure mode b. Effects	Potential solutions: : a. Prevention b. Protection c. Other actions d. Remarks

Figure 3 : Example of FMECA array

Types of errors For an actor of the external device type, exchanged messages correspond either to operations either to signals. Each error model from E.1 to E.10 can be used for these messages. As an example, for the message *Control movements(parameters)* of figure 4, errors can consist in sending wrong parameters (error E.6, E.7 and E.8), they can be caused by delays (E.5 et E.10) or errors from the *master site* during the message emission (E.1, E.2, E.3, et E.4). For this type of actor, it is important to notice that the error types include the failures of the sender object, but also the failures of the link with the receiver object (E.11). This allows to consider the telecommunication aspects of some kind of application as telemedicine.

Failure mode analysis The modeling of all exchanged messages with sequence diagrams allows to process to an analysis very soon in the development process. It leads to formulate safety and reliability requirements from the start, with no need to develop the design choices to make components communicate. For example, on the diagram figure 4, the type of communication is not specified (intranet, internet, RS232, etc.), but failures of messages from the master site can be considered anyway.

Figure 5 shows a generic analysis of the failure mode (E.6, E.7 or E.8 types) of the message *Control movement(parameters)* of the sequence diagram on figure 4.

Then, from this analysis, one can propose a solution for this failure mode by the modification of the system UML diagrams. The solution we propose consists in temporarily changing the robot controller state. This is illustrated on the state diagram of figure 6.

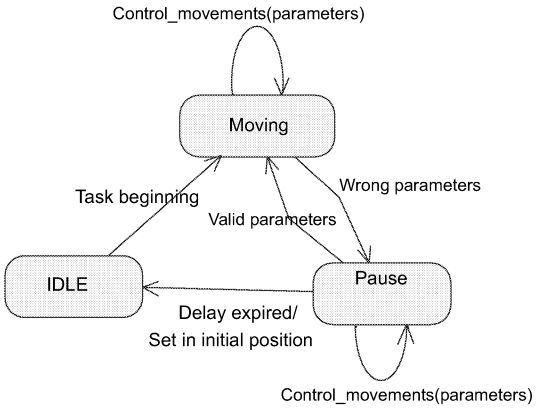


Figure 6 : State diagram taking into account a failure mode of *Control movements()*

7. Conclusions and perspectives

Considering the growing system complexity and the urgent need to take safety into account, we proposed an approach based on risk analysis. We considered that damages appear into a system because of its dynamics. Thus, hazards are partially linked up with message exchanges into a system. Going on the notion of *Message* in UML, we presented some models of errors related to this concept. We showed that these models can also be useful in other fields (we validated it on the design of a highly critical telemedicine application with robots [8]). Moreover, in other papers, we demonstrated that these models was generic and thus could be used for different applications. The link between the object-oriented concepts and the FMCEA technique has been established, although the FMECA was originally dedicated to functional analysis. The approach we propose is integrated into a whole development process based on the UML notation where the risk analysis is developed in parallel using the same UML models. The next technical step would be the development of tools to automatically integrated FMECA to UML design diagrams. Our theoretical perspective is to apply the same philosophy not to FMECA but to fault tree technique.

References

[1] F. Bitsch. Requirements on methods and techniques in perspective to approval process for railway systems. In *Second International Workshop on Integration of Specification Techniques*

Date Author :			Project name						Page : /
Code	Message	Failure mode (error)	Cause	Effects a. Same level b. Upper level c. System level	Risk Severity	Probability	Risk	Possible detection means: (on line) a. Failure mode b. Effects	Potential solutions: : a. Prevention b. Protection c. Other actions d. Remarks
	Control movements (parameters)	Bad parameters (error E.6 to E.8)	Master site failure or link failure	Unwanted movement	1	P	H	Parameters coherence verification (digital filter)	a. Use of a standard protocol b. Ignore value, set the controller in waiting state

Figure 5 : A failure mode analysis of the message *Control movements()*

for Applications in Engineering (INT 2002), Grenoble, France, April 2002.

[2] A. Bondavalli, M. Dal Cin, D. Latella, I. Majzik, A. Pataricza, and G. Savoia. Dependability analysis in the early phases of UML based system design. *International Journal of Computer Systems - Science & Engineering*, 16(5):265–275, September 2001.

[3] A. Bondavalli, I. Majzik, and I. Mura. Automated dependability analysis of UML designs. In *2nd IEEE International Symposium on Object-Oriented Real-time Distributed Computing (ISORC'99), Saint Malo, France*, pages 139–144. IEEE Computer Society Press, 1999.

[4] A. Bondavalli, I. Majzik, and I. Mura. Automatic dependability analysis for supporting design decisions in UML. In *4th IEEE High Assurance System Engineering Symposium (HASE99) Washington D.C., USA*, pages 64–71. IEEE Computer Society Press, November 1999.

[5] G. Booch, J. Rumbaugh, and I. Jacobson. *Unified Modeling Language Users Guide*. Addison Wesley Longman, 1999.

[6] J. Górski and B. Nowicki. Object oriented safety monitor synthesis. In D. Gritzalis, editor, *Third International conference on reliability, quality and safety of software intensive systems (ENCRESS'97), Athens, Greece*, pages 121–133. Chapman and Hall, May 1997.

[7] J. Górski, B. Nowicki, and A. Wardzinski. Holistic and partial system models in safety analysis. In *International conference on probabilistic safety assessment (PSA'96), Park City, USA*, volume 2, pages 1301–1309, September 1996.

[8] J. Guiochet. *Safety management of service robot systems - UML approach based on system risk analysis (in french)*. PhD thesis, Institut National des Sciences Appliquées de Toulouse, 2003.

[9] J. Guiochet, B. Tondu, and C. Baron. Integration of UML in human factors analysis for safety of a medical robot for tele-echography. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Intelligent Robots and Systems for Human Security, Health, and Prosperty IROS 2003*, October 2003. accepted.

[10] J. Guiochet and A. Vilchis. Safety analysis of a medical robot for tele-echography. In *Proc. of the 2^{nd} IARP IEEE/RAS joint workshop on Technical Challenge for Dependable Robots in Human Environments, Toulouse, France*, pages 217–227, October 2002.

[11] IEC 61508. Functional safety of electrical/electronic/programmable electronic safety-related systems. International Electrotechnical Commission, 2001.

[12] ISO 14971. Medical devices - Application of risk management to medical devices. International Organization for Standardization, 2000.

[13] ISO/IEC Guide 51. Safety aspects - Guidelines for their inclusion in standards. International Organization for Standardization, 1999.

[14] P. Johannessen, C. Grante, A. Alminger, U. Eklund, and J. Torin. Hazard analysis in object oriented design of dependable systems. In *2001 International Conference on Dependable Systems and Networks, Göteborg, Sweden*, pages 507–512, July 2001.

[15] J-C. Laprie. *Dependability: Basic concepts and terminology in English, French, German, Italian and Japanese*, volume 5 of *Dependable Computing and Fault Tolerance*. Springer-Verlag, Austria, 1992.

[16] I. Majzik and A. Bondavalli. Automatic dependability modeling of systems described in UML. In R. Chillarege, editor, *9th International Symposium on Software Reliability Engineering (ISSRE'98), Paderborn, Germany*, vol-

ume 2, pages 4–7. Th. Illgen, IEEE Computer Society, November 2000.

[17] MIL-STD-1629A. Procedures for performing a Failure Mode, Effects and Criticality Analysis. Military Standard, 1980.

[18] B. Nowicki and J. Górski. Object oriented safety analysis of an extra high voltage substation bay. In W. Ehrengerber, editor, *SAFECOMP'98*, pages 306–315. Springer-Verlag, 1998.

[19] OMG. OMG Unified Modeling Language Specification v1.4. Technical report, Object Management Group, September 2001.

[20] OMG. 2nd revised submission to OMG RFP ad/00-09-02 - Unified Modeling Language : Superstructure - version 2.0. Technical Report ad/2003-01-02, Object Management Group, January 2003.

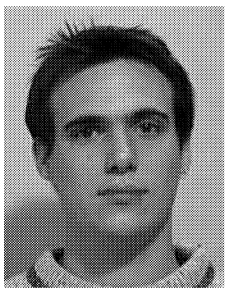
[21] OMG. OMG Unified Modeling Language Specification v1.5. Technical Report formal/03-03-01, Object Management Group, March 2003.

[22] G.J. Pai and J.B. Dugan. Automatic synthesis of dynamic fault trees from UML system models. In *Proceedings of the 13th International Symposium on Software Reliability Engineering (ISSRE'02)*, 2002.

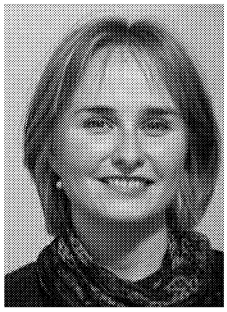
[23] Y. Papadopoulos and M. Maruhn. Model-based automated synthesis of fault trees from matlab-simulink models. In *Int. Conf. on Distributed Systems and Networks (DSN'2001), Gothenburg, Sweden*, pages 77–82, 2001.

[24] S. Yacoub, H. Ammar, and T. Robinson. A methodology for architectural-level risk analysis. In *11th International Symposium on Software Reliability Engineering (ISSRE'2000), San Jose, CA*, pages 210–221, October 2000.

Biography



JEREMIE GUIOCHET studied electronics and computer sciences at ENSEIRB, a French engineering school based in Bordeaux. He obtained his degree in 1998 and came in Toulouse to start his research in the LESIA laboratory. He received the Ph.D. in 2003 from the INSA, Toulouse, France. He is now an associate professor in Computer Science in University of Toulouse III. He does his research both in LESIA and in GRIMM/ISYCOM. His research interests are in the areas of software modeling and safety critical systems and particularly medical robots.
Contact : guiochet@univ-tlse2.fr
http://www.lesia.insa-tlse.fr/ guiochet



CLAUDE BARON was born in Perpignan, France, and went to the National Institute of Applied Sciences of Toulouse to study computer sciences. She obtained her degrees in 1992 and her PhD in 1995. She is now an assistant professor at INSAT where she teaches real time systems and system reliability. She does her research at the LESIA laboratory where she is in charge of the SFS group.
Contact: claude.baron@insa-tlse.fr
http://www.lesia.insa-tlse.fr/francais/membres/p-pers/baron-claude/index.html

OBJECT-BASED REMOTE RESOURCE ACCESS FRAMEWORK

Marin Lungu, Constantin Pistol and Anita Lungu
Computer Dept., Faculty of Automation, Computers and Electronics, University of Craiova
Str. Lapus Nr. 5, Craiova, Romania
Marin.Lungu@comp-craiova.ro
costi@xts.ro
anita@xts.ro

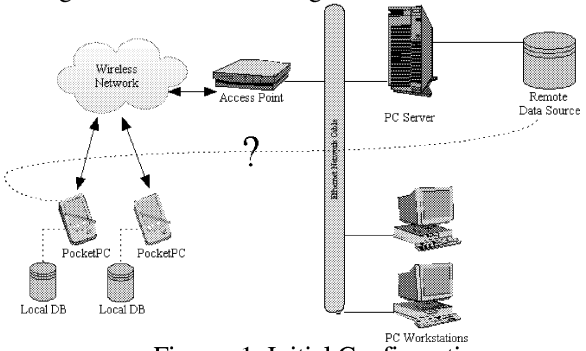
KEYWORDS
Communication, framework, mobile, object, data access.

ABSTRACT

In today's mobile solutions market there can be identified a paramount need for a platform independent communication technology that is in the same time open, secure and based on standards. This paper will present the architecture of a versatile object oriented mobile application middleware whose remote data access infrastructure incorporates XML web services for mobile devices and addresses all the above requirements.
The framework presented provides resource limited mobile terminals the means for accessing outside (remote) data in a simple, yet flexible way. This remote data can come from different sources (remote databases, applications, etc.), and the framework will offer an uniform, object-based way of accessing it. (Pseudo) Distributed Transactions and Page Based data retrieval are implemented for using a DBMS as a data source.

INTRODUCTION

The framework developed (named DAX - Data Access using XML) is targeted (mainly) at the following scenario. The terminals have an active wireless link to a cable based Ethernet PC network. On this network the main database resides. This database makes available at least one standard data provider (ODBC, ADO, etc.). Database access from the PocketPC (mobile terminal) is required. This would have been a non-issue have database drivers existed for WindowsCE – but such drivers are unavailable. Database functionality on the PocketPC is available, however only at a local level. A sample system configuration is the following:



Figures 1: Initial Configuration

As presented before, the PocketPC handhelds have access to a local database, as well as to the PC network (by means of a wireless access point/bridge). There is however no standard way of accessing the remote database.

DEFINING DATA ACCESS

In the simplest form, we can consider data retrieval as a two step communication:

- the mobile application (running on the PocketPC) makes a data request call to the framework, passing as a parameter the command that must be run against the remote data source (usually an SQL command) .
- the data request call returns a list of records - the result of executing the command against the remote data source.

To provide such functionality, the framework should make available to the PocketPC applications a function with a prototype similar to the following:

List<RecordType> GetData(CommandType cmd) ;

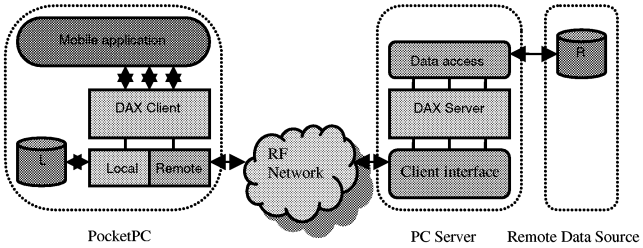
This covers the general case of data retrieval - but in many cases it is useful to have specific functions that relate to a particular data source that will be used. The framework implements a more advanced object-based concept.

DAX FUNCTIONAL MODULES

Since the PocketPC is unable to access a remote data source directly, an intermediate PC server must be introduced in the communication framework. The PocketPC part of DAX will then forward to it the *GetData()* request call (together with the provided command). The PC part of DAX, running on this server (DAX Server) will do the actual work, connecting to the data source, running the command against it and then sending back the result (data) to the PocketPC component.

DAX framework also integrates local database connectivity, using the same interface as for remote access. A flag is set by the mobile application signaling whether local or remote data access is desired; other than that, the actual use, as far as the application is concerned, is identical.

The logical design of such a framework is depicted in the following figure:



Figures 2: DAX logical design

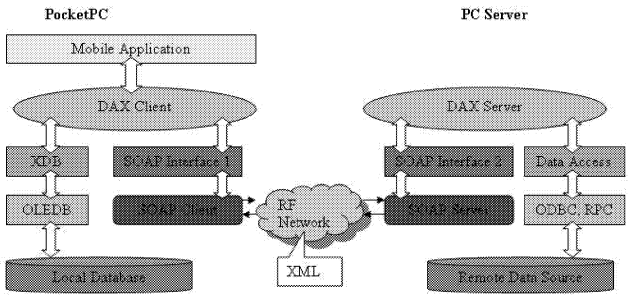
It's worth noting that the Remote Data Source needs not be (and usually isn't) local to the PC Server. As long as it is possible to build a proper Data Access interface for it, it can be integrated in the framework. This Data Access interface can be based on completely different technologies, depending on the type and location of the data source. If it is a remote database with available drivers, the interface can be built around ODBC, ADO or similar standards. If it is a remote application/server (for instance an industrial control system that can be queried for data) then this interface might be developed around Remote Procedure Calls (RPC) or OLE for Process Control (OPC).

DAX TECHNICAL MODULES

Since DAX has client/server architecture, with the client and server running on different platforms and different operating systems, one major decision is the choice of communication protocol and data representation. These will be used for connecting DAX Client with DAX Server.

After looking at current standards for communication, the decision was to use Simple Object Access Protocol (SOAP) for communication and XML for data representation.

The dataflow in DAX Framework looks like this:



Figures 3: Dataflow

Given the choice of protocol, it follows that DAX Server has to run on top of a SOAP server – or otherwise implement its own. The platform chosen was Microsoft's ASP.NET and the developing environment used was Microsoft Visual C#.

On the PocketPC side, the DAX Client (and more precisely its Remote module) needs a SOAP client. The freely available library PocketSOAP (www.pocketsoap.com) was used for this purpose. It presents itself as an open source SOAP client (packaged as a PocketPC COM component), with a straight forward, easy to use interface for SOAP based communication. The package includes a HTTP 1.1 transport for making HTTP based SOAP requests.

XML PARSING

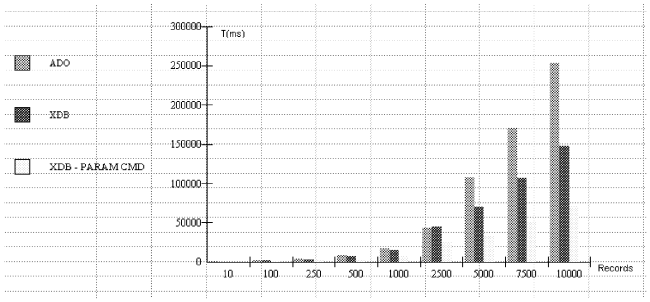
The PocketSOAP's built-in DOM-style XML parser was removed in favor of a custom built SAX-style parser. The reason was memory consumption on large XML files – DOM type parsers create a full in-memory tree representation of the XML. This tree representation has memory requirements of at least one magnitude higher that the space consumed by the XML representation.
The custom DAX Parser goes through the XML sequentially, creating directly a list of records without the intermediary tree representation.
The core of the parser is the Boyer-Moore algorithm, one of a larger family of algorithms designed for performing exact string searches, approximate string searches, 'sounds-like' string searches, and other types of textual comparisons [1].

POCKET PC LOCAL DATABASE ACCESS

Microsoft offers a standard database access layer for their Windows CE operating system in the form of ADO CE (ActiveX Data Objects for CE). ADO CE proved to be adequate for a small database activity, but under medium to heavy use it failed to deliver a good level of performance reliability. Memory leaks within ADO CE's code were causing major application slowdowns, and the performance itself (in terms of speed) was disappointingly low. An alternative had to be found.

So, a new interface called XDB was created as a wrapper around OLEDB (a lower level layer). It provides the ADO CE functionality DAX Framework needs, and it improves by adding parameterized query support and exception-based error handling. The interface XDB creates is designed along the lines of ADO.NET, with separate Connection, Data Provider, Data Reader and Command classes provided.

The boost in performance gained by switching to XDB is significant. An insertion test with records containing 3 strings and 1 integer in a table with no indexes defined is presented. The test configuration was a Symbol PPT 2800 PDA, with Windows CE 3.0, using ADO CE 3.1 and SQL Server CE 2.0. It yielded the following results:



Figures 4: Database insertion speed

The yellow bars show the speed XDB has when using parameterized queries.

DAX SERVER: MAINTAINING STATE

When presenting the system's functional specifications, DAX was considered as a provider on the PocketPC of a series of functions dealing with data access. While this API-like approach works well for explaining the principles behind the framework, it has some problems.
Functions of the *GetData()* or *Execute()* type (in the context of database access) are a perfect example of API-style functionality. They work well that way, and a more advanced design is not needed.
Considering functions such as *BeginTransaction(), Commit(), Rollback()* an implicit property is observed: they need to maintain state. If later on, after a *BeginTransaction(),* a call on *Execute()* is made, DAX Server must be able to recognize that this latter call must be executed in the transaction context initiated by the former call. This also has the side-effect that separate connections to the remote database must be maintained (by the server) for each client.

REMOTE ACCESS: OBJECTS AS PRIMITIVES

DAX solves this by using (from a logical point of view) objects as primitives instead of function calls [2].

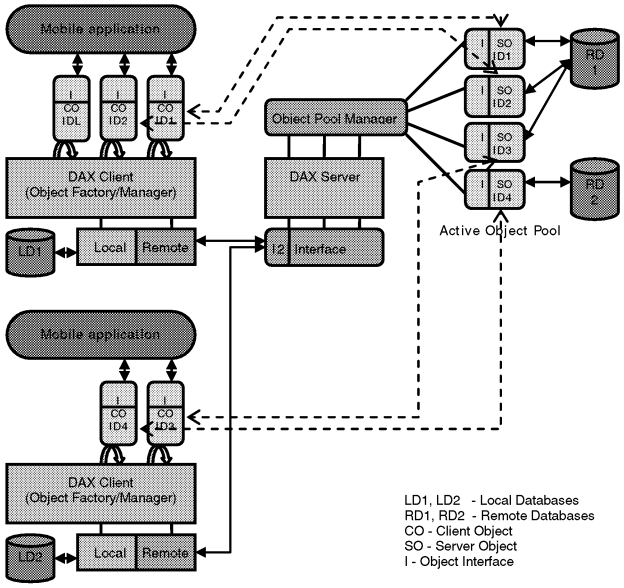


Figure 5: Object based design

From the point of view of the Mobile Application, DAX Client acts as an Object Factory, delivering client objects (CO) capable of data access to remote (and local) databases. These objects present to the application an interface I, containing the desired functionality. In the case of database access, this public interface looks like the following:

```
class DAX_Interface{
  virtual bool Open(string sConnectionString);
  virtual bool Close();
  virtual List<RecordType> GetData(string sSQLCmd);
  virtual bool StartTransaction();
  virtual bool Commit();
  virtual bool Rollback();
  virtual bool Dispose();
}
```

The functions are self explanatory. *Dispose()* takes care of destroying the CO and it's corresponding SO.

The actual interface provided by DAX has additional functions for:
- calling stored procedures
- executing parameterized queries
- supporting page-based query result retrieval

SERVER AND CLIENT OBJECTS

From the previous paragraphs, it can be seen that the Client Objects (CO) that DAX Client provides can not accomplish their advertised functionality on remote databases by themselves, as they are local to the client. Only the server can actually connect to the remote database and resolve the calls.

To solve this, the server creates its own version of Client Object – a Server Object (SO). A SO has the same interface as CO, but there is one crucial difference: because Server Objects reside on the server, they actually have the capability to connect to the remote database and resolve the calls in their interface.

When DAX Client is asked to generate and deliver a CO for the Mobile Application, DAX Server is informed and asked to generate a corresponding SO. A function with the prototype:

String CreateObject(string sConnectionString);

is available in the server's interface (I2) for this purpose. What it does is to create a SO, connect it to the remote database specified by the connection string, give it a unique ID [2] (a Global Unique Identifier –GUID- is used) and return this ID to the DAX Client that made the call. At this point, DAX Client creates a CO and gives it the ID received from server. By doing so, a logical one-to-one mapping is established between the CO delivered to the application and it's corresponding SO object on the server. All future calls on this CO made by the Mobile Applications are trapped by DAX Client and forwarded to DAX Server together with the object ID. The SO with that ID makes the specified call, and data is returned and passed back through the layers until it reaches the Mobile Application. As far as the application is concerned, the CO has resolved the data call – all the communication underneath it is completely transparent.

FOLLOWING AN USE CASE

As an example, the *GetData()* function is being called by the application on one of it's Client Objects (presumed with ID2). Here are the steps that take place on such a call:

- Mobile Application calls *GetData(SQLCmd)*
- DAX Client traps the call and contacts DAX Server's I2 interface, calling *GetData(SQLCmd, ID2)*. To do that, both parameters are serialized to XML and packaged in a SOAP call message[3].

- DAX Server receives the call, asks the Object Pool Manager for the SO with id ID2 and calls *GetData(SQLCmd)*on that object (as both CO and SO implement interface I).
- The data received from the function call is serialized to XML and sent back to DAX Client as a SOAP response message.
- DAX Client deserializes the data using the XML Parser and builds the list of records the application expects.
- CO's *GetData()* call return to the application with the list of records as a return value.

It can be seen that I2, the functional interface the server provides is very similar to I, the interface of the CO/SO. The difference comes from the fact that I2 must be able to give access to all the SO that the server maintains. This causes function from I2 to have an additional parameter: the object ID – which specifies the exact SO that will execute the function. Object Factory-type functions are also provided by I2 (like *CreateObject(objID)* and *Dispose(objID)*.

OBJECT POOLING

Given this design, DAX Server must maintain a pool of active objects. Since DAX Server is built as a set of state-less WebServices, a separate state-full server Object Pool Manager is implemented. Situated outside the web server's process (but still inside the .NET virtual machine), this manager is called by the WebServices when they require specific Server Objects to be created, called or destroyed.

As figure 6 shows, there are actually two pools that are maintained by the Object Pool Manager. The first is the Active Object Pool – the one that keeps all the Server Objects in use by the clients. The other is the Free Object Pool, and its sole purpose is to optimize performance under heavy load.

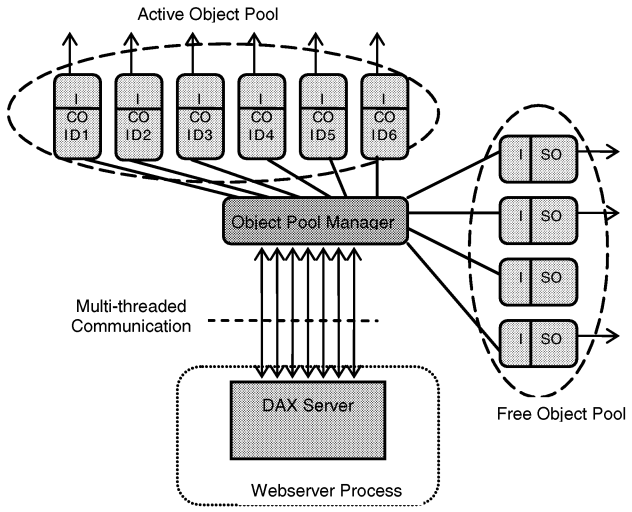


Figure 6: The two object pools: in-use (active) and free

When the Object Pool Manager receives from DAX Server a request to dispose (delete) a certain active Server Object, instead of deleting it the manager will put it in the Free Object Pool. If the object had an open connection at the time of the disposal request, the connection is kept open.

The reason this approach improves performance (at the cost of memory consumption) is that when an object creation call is received, the manager will scan the pool of free objects for one that has a matching connection string. If it is found, then this object gets a new ID and it is transferred to the Active Object Pool. In this scenario, time is gained because we skip the object creation and database connection initialization.

If no matching connection string is found, a connection-less object is used – avoiding object creation.
There is some background work the Object Pool Manager must do in order to keep things running smoothly. Depending on the amount of free server resources it must trim down the free pool, disconnect certain connected free objects when the number of active connections gets too high, and so on.

Because DAX Server is multithreaded, the manager must be thread-safe. It must be able to do all operations on the shared pools properly in this multithreaded scenario. The current implementation uses the principle of critical sections, mainly with the help of the *System.Threading.Monitor* included in the .NET Framework standard classes[3].

DAX AND DCOM

Looking at the architecture employed by DAX, one could notice similarities to the DCOM model. DAX Client and DAX Server are basically object brokers for calls originating from client, interface-only, objects and executing on actual server objects. There are many differences of course, and DAX is nowhere near to being a DCOM replacement. It has however certain advantages. Being based on SOAP and XML, DAX Client can be ported on most of the current platforms, if such need arises. As long as it respects the defined interfaces and protocols for communicating with the server, the rest is local implementation detail and does not affect the system. To give an example, the PocketPC on which DAX Client is implemented does not support DCOM – yet with this framework one can have some of its benefits.

DISTRIBUTED TRANSACTIONS

The architecture of DAX allows for (pseudo) distributed transactions. It can be considered an application that starts a transaction, does some work, and then sends the ID of the object on which it worked to another application, on another computer. The DAX Client can create a new Client Object with that ID, without creating a new SO (it's already on the server). In the same manner, the ID can be passed so that several users/computers do work on the same transaction. In the end, a final application can decide whether to commit the transaction, or rollback the changes done by all the other computers. This isn't exactly a distributed transaction (the transaction never actually moved, belonging all-along to a single Server Object in

DAX Server). But the effect, as far as the client applications are concerned, is the same.

CLIENT OBJECTS FOR LOCAL ACCESS

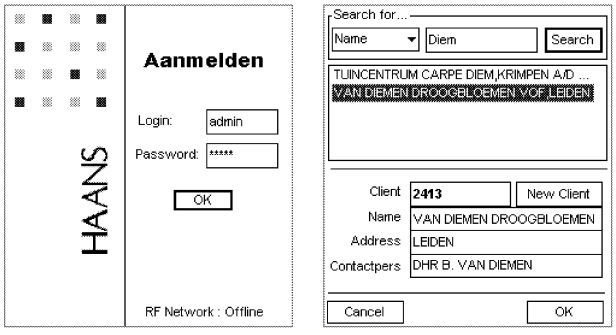
Returning to the Client Objects / Server Objects – what happens when the application demands the creation of an CO for accessing the local Pocket PC database? There isn't a great amount of difference from the case of objects for remote access. The main point is how the calls to it are handled by DAX Client: instead of passing the call to the Remote module for sending to server, the call is handled by the Local module – which will access the local database and resolve the call. This also means the object creation is local – there is no need for a corresponding Server Object, and neither is an ID needed.

SAMPLE APPLICATION

The framework was originally designed for and implemented in a large SFA (Sales Force Automation) application targeted at the PocketPC platform. The company contracting it has a large Navision ERP (Enterprise Resource Planner) which makes available a standard ODBC driver for data access.

The application itself is designed mainly for order taking. The functionality and application workflow is moderately complex, but they will not be presented as it is out of the scope of this paper. There are presented only the technical features in which the use of DAX had a direct impact:

- fast data sync with the remote server by using parameterized queries and callback insert functions.
- complete online-only mode : Navision is accessed as if it would be a local database
- complete offline-only mode : the local database (that was previously synced) acts as the main database. When the terminal enters online mode, the changes made are quietly sent to Navision.
- fault tolerance : if a client works online for half of an order, then suddenly the connection to Navision is lost, the data framework is capable of continuing without notifying the user. Later on, when connection is back online, the two halves of an order (remote and local) are assembled back into one logical entity and sent to Navision , canceling the first (unfinished) records.



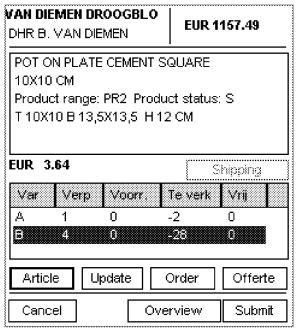


Figure 7: Application Screenshots

CONCLUSION

DAX Framework was mainly presented in terms of accessing remote databases. In this field, it does offer data access and capable data sync functionality. But the idea behind it, of having Server Objects doing actual work and Client Objects presenting to an application only the interface is suitable for many other uses.

DAX could be used for remote control of outside processes/applications/etc. Terminals that would otherwise be ruled out because of the lack of resources or direct connectivity can then be used for such purposes (the PocketPC is one example, but there are many others).
Monitoring processes from terminals that do not have drivers to read data or lack direct access to the data source because of other factors is also made possible.

An interesting use is that of resource off-loading. For instance, in an application that is doing heavy mathematical computations on terminals that have limited resources can benefit from it. Server Objects can be design to offer an interface for doing these computations. When the application is offline, local resources are used to perform the computations, and things will go slower. But when online, the calls are forwarded by DAX to Server Objects, which presumably run on machines that are less resource limited. Thus, the computations are actually performed outside the client and faster.

DAX is general enough to be used on a wide range of applications, and the fact that it is based on open standards such as XML and SOAP make it an option on a wide range of hardware and software platforms as well.

REFERENCES

[1] Alfred V. Aho (1990) *Algorithms for finding patterns in strings*. [2] Xiangyang Liu *Component to Access databases in the .NET Environment* [3] James Snell, Doug Tidwell, Pavel Kulchenko (2002), *Programming Web Services with SOAP*, O'Reilly & Associates, Inc.

SIMULATION TOOLS

SIMULA

OBJECT ORIENTED MODULARITY OF SIMULA

Jaroslav Sklenar
Department of Statistics and Operations Research
University of Malta
Msida MSD 06, Malta
Web: http://staff.um.edu.mt/jskl1/
E-mail: jaroslav.sklenar@um.edu.mt

KEYWORDS
SIMULA, Object-oriented programming, Modularity, Discrete simulation.

ABSTRACT

The paper starts with a brief summary of basic facts about SIMULA with special emphasize on those features of the SIMULA's object model that are different compared with other object-oriented languages. Main classes that can contain local nested classes together with the block prefixing mechanism represent modularity achieved by the object model itself without any additional definitions that are needed in other object-oriented languages. One of the drawbacks of SIMULA is some unbalance between its sophisticated object model and user-friendly time control of the system class Simulation on one side and lack of some simulation supporting facilities like collection of statistics on the other side. To improve the situation and to demonstrate work with SIMULA main classes, two main classes supporting transparent statistics and advanced queues have been created. Their basic ideas are explained together with an example application in a simple queueing model.

INTRODUCTION

SIMULA is the first Object-oriented language (OOL) that introduced all important ideas of object-oriented programming (OOP) in 1967, long time before the OOP boom in late eighties occured. Let's start by some myths that are still around. First myth: "*SIMULA is a discrete simulation language*". This may be caused by the fact that there used to be two versions of the language: SIMULA I (Simulation Language) was a process-oriented discrete simulation language. SIMULA 67 (Simple Universal Language) is a general OOL that without its system class Simulation has nothing to do with simulation what so ever. For example the book (Kirkerud 1989) does not deal with simulation at all. SIMULA I has not been used for a long time, so today there is no need to make the difference between the two versions (SIMULA = SIMULA 67). For more details about the history of the two SIMULAs read the article (Holmevik 1994) that thanks to Mr. Holmevik's kind permission is available on-line on author's page at http://staff.um.edu.mt/jskl1/simula.html. The basic facts about SIMULA, its system classes and related interfaces are shown in Figure 1 taken from the text (Sklenar 1997).

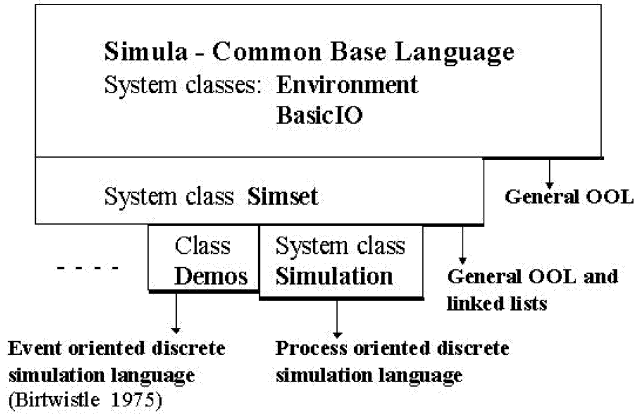


Figure 1: Main Parts and Interface Lines in the SIMULA Environment

Second myth: "*SIMULA is basically ALGOL 60 plus simulation extension*". This nonsense may be caused by the fact that SIMULA uses ALGOL 60 syntax and its block structure. This is logical because in the late sixties ALGOL 60 was a de-facto standard in high-level programming. Today it is obvious that due to its OOP capabilities SIMULA is as far from ALGOL as ALGOL is far from the assembly language.
SIMULA has been criticized that on one side there is a high-level sophistication of the language itself combined with rather limited simulation capabilities except time control on the other side. For example in the book (Bratley et al. 1987) there is the following assessment of SIMULA: "*Accumulating statistics in a simulation (unless Demos is used) is as tedious as in Fortran, and producing formatted reports is worse. The combination of well nigh impenetrable complexity in certain areas with a tedious triviality in others makes Simula a curiously ill-balanced tool.*" This criticism is more or less true. Lack of supporting simulation tools like transparent collection of statistics, more advanced queues than the basic lists available in the system class Simset, report generation, etc. is caused by the fact that the language has been formally frozen in 1968. This decision was controversial. The advantage is that there is only one SIMULA with minimal differences between implementations. On the other hand future development was made difficult. SIMULA is thus more an OOP background for the creation of simulation tools than a complete user-friendly simulation environment. Probably the best known one of such tools is DEMOS (Birtwistle 1979). Some classes described later in

this paper try to partially fill the gap between the standard system class Simulation and the large, user-friendly, but not so easy lo learn DEMOS.

MODULARITY IN SIMULA

All non-trivial programs have to be divided into relatively small manageable parts – modules, also called libraries. The terminology is not unified. Turbo Pascal libraries are called units, C++ has header files, Java libraries are called packages, etc. SIMULA is probably the only language where libraries (packages) are implemented by the object model itself without any additional definitions. Packages in SIMULA are represented by the so-called *main classes* that can contain their local nested classes. SIMULA classes consist of up to four parts in the following lexicographical order:

Parameters
Properties (Attributes)
Methods
Life (Body)

First note that in most OOLs there exist only properties and methods. Parameters are basically properties initialized when the instance is created. So the most important difference between instances of SIMULA classes compared with instances of other typical OOLs like for example Java is the existence of code whose execution starts when the instance is created. So instances can in fact be co-routines. This is obviously used in the system class Simulation where instances of the class *process* represent parallel processes of the model. SIMULA classes also do not have constructors. Parameters are initialized as a part of instance creation, more complicated initialization can be done as the first part of the life rules as shown in the following code.

```
Class Rectangle (Width, Height);
   Real Width, Height;
              ! Class with two parameters;
Begin
   Real Area, Perimeter;   ! Attributes;

   Procedure Update;       ! Methods;
    Begin
      Area := Width * Height;
      Perimeter := 2*(Width + Height)
    End of Update;

   Boolean Procedure IsSquare;
      IsSquare := Width = Height;

! Life of rectangle started at creation:;

   Update;
   OutText("Rectangle created: ");
   OutFix(Width,2,6); OutFix(Height,2,6);
   OutImage

End of Rectangle; ...
```

```
Ref(Rectangle) R; ...
R :- New Rectangle(50, 40); ...
```

Nevertheless in our context the most important fact is the following one: attributes can be classes with the same above structure. Terminological note: properties are typically interpreted as value attributes of instances while methods are their procedure attributes. In SIMULA context an attribute can be a simple value, but it can be a complex class with local classes. Local classes can be used to represent knowledge about various data structures and other real or abstract parts of a certain package. The package is then represented as a main class that can generally be instantiated, used as a super-class in inheritance trees, etc. As examples of main classes see the description of system classes Simset and Simulation for example in the book (Pooley 1987) that is also available electronically, or the "geometry" example in the text (Sklenar 1997).

There is another difference between SIMULA and other OOLs, the so-called block prefixing. The above code shows that instances of SIMULA classes are created in a way that has been accepted by most OOLs. Just note the special assignment operator used with reference variables (pointers do not exist in SIMULA) and the explicit initialization of the attributes. But there is another possibility how to create instances:

```
Rectangle(10,20) Begin … End;
```

Passing through the *begin* of a prefixed block creates an unnamed class instance. As at the standard instance creation the parameters are initialized and the life code is started. In this example the life terminates after performing the OutImage (writeln) statement, the next statement is the first one of the prefixed block. By using the statement *inner* the life of the instance can be interrupted by passing control to the prefixed block. After its last statement the life is resumed. This is useful for various cleanup operations. So what is the use of this mechanism? In this example the effect is the same as if the block contained declaration of four additional local variables, two of them being initialized, declaration of two local procedures and life statements at the beginning of the block. Of course all this is not very useful. The mechanism is very useful when a block is prefixed by a main class. The effect is the import of all declarations from the main class that may represent thousands of lines of code. For example prefixing by the system class Simulation imports process-oriented discrete simulation capabilities:

```
Simulation Begin … End;
```

Note that SIMULA does not need any package importing statements like *uses* of Turbo Pascal or *import* of Java. Next two main classes intended as a simulation support will be described.

TRANSPARENT STATISTICS

Probably the first user-friendly transparent collection and computation of statistics has been implemented in the SIMSCRIPT language by using the so-called *left monitoring* of selected variables. It is based on the idea presented by the paper (McNeley 1968) where it is called *store association*. The idea is simple, an assignment statement $v = e$ evaluates the expression e, the value is passed to a routine associated with the variable v, the value returned by the routine is stored to v. For transparent statistics the value is not modified, the routine just updates the figures needed for the selected statistics on the variable. Obviously this mechanism cannot be used directly as it is built-in in the language. The OOP solution suggested by O-J Dahl during a discussion following the McNeleys presentation is based on using an updating method. A statistically observed variable is then a class instance whose one property is the actual value. Other properties together with the methods do the rest as shown in the following code that outlines the declaration of the main class *statsim* as a subclass of Simulation.

```
simulation class statsim;  begin

 class accumulator(x); real x;
 ! time statistics object, x=initial value;
  hidden valuex,sumsq,inittime,lasttime;
  begin
   ! attributes:;

   real valuex, maxvalue, minvalue, sum,
        sumsq, inittime, lasttime;

   ! methods:;

   procedure initiate(initx); real initx;
   begin
     valuex := initx;  maxvalue := initx;
     minvalue := initx;
     sum := 0; sumsq := 0;   ! integrals;
     inittime := time;  lasttime := time;
     ! initialisation and last update time;
   end;

   procedure updateto(newx); real newx;
   begin
     real d;
     d := (time - lasttime)*valuex;
                  ! integral increment;
     sum := sum + d;
     sumsq := sumsq + d*valuex;
                  ! updating integrals;
     lasttime := time;  valuex := newx;
     if (newx>maxvalue) then
           maxvalue := newx;
     if (newx<minvalue) then
           minvalue := newx;
   end;

   procedure updateby(delta); real delta;
   begin  updateto(valuex + delta)  end;

   real procedure average;
   begin
     if (time>inittime) then begin
       updateby(0);
       average := sum/(time - inittime)
     end else
       average := 0
   end;

   real procedure variance;
   begin
     if (time>inittime) then begin
       real a; a := average;
       variance := sumsq/(time-inittime) -
                   a*a
     end else
       variance := 0;
   end;

   real procedure stddev;
   begin  stddev := sqrt(variance)  end;

   initiate(x);  ! life of accumulator;
 end of accumulator; ...
end of statsim;
```

Accumulator instances are used for variables with time-based statistics (length of a queue). Another two classes *tally* and *histotally* are declared for time-less statistics (waiting time in a queue) with or without histogram collection. The following text has been written to a text file by the method *fileout* of a *histotally* object used to collect statistics on time spent by customers in a queueing system (the histogram has been truncated to fit the column width).

```
Time spent in system statistics:
Average      8.20
Std Dev      2.08
Minimum      0.62
Maximum     16.25
Updates      4957
   X      PDF    CDF    Histogram
 1.07   0.001  0.001
 2.13   0.001  0.002
 3.20   0.005  0.007   *
 4.27   0.021  0.028   *****
 5.33   0.054  0.083   **************
 6.40   0.109  0.192   **********************
 7.47   0.175  0.366   **********************
 8.53   0.199  0.566   **********************
 9.60   0.194  0.760   **********************
10.67   0.128  0.888   **********************
11.73   0.065  0.953   ****************
12.80   0.031  0.984   ********
13.87   0.010  0.994   ***
14.93   0.004  0.998   *
16.00   0.001  1.000
16.25   0.000  1.000
```

Another method saves the cdf distribution table that can be later loaded by a method of a random object and used in other simulation models.

ADVANCED QUEUES

The system class Simset implements two-way linked circular lists. These can be used directly for FIFO and LIFO queues without any kind of statistics. That's why another main class called *quesim* has been declared. It is a subclass of *statsim* with classes representing a customer, and FIFO, LIFO and priority queues. All usual statistics is collected including waiting costs of customers. A detailed description of the classes is beyond the scope of the paper, so only the classes together with parameters, attributes, methods, and life rules are listed:

process class ***g_customer*** (general customer)

parameters:
priority (customer's priority)
waitcost (waiting cost per unit time)

attributes:
waitingcost (total waiting costs till now)
queuetime (last queueing time)
waitingtime (total waiting time till now)
timeinq (time of entering a queue)
timeins (time of entering the system = creation time)

methods:
initiate (initialization)
waitin(Q) (entering a queue of any type)
nowaitin(Q) (moving through queue without waiting)
systemtime (time spent so far in the system)

life: initialization, rest supposed to be defined in subclasses.

head class ***fifo*** (FIFO queue with statistics)

parameters:
qname (name of the queue for report heading)
capacity (used for limited queues, *maxint* if unlimited)

attributes:
isize (current length - hidden)
length (length statistics)
waiting (waiting time statistics)
busy (busy time statistics)
waitingcost (total waiting costs till now)
arrivals (number of arrivals)
nwarrivals (number of not waiting arrivals)

methods:
initiate (initialization)
enqueue(C) (enqueueing a customer)
gothrough (customer moves through, no waiting)
takefirst (removes and returns the first customer)
LQ (current length)
LQaverage (average queue length)
LQmaximum (maximum queue length)
LQstddev (standard deviation of queue length)
WQaverage (average waiting time)
WQnzaverage (average nonzero waiting time)
WQmaximum (maximum waiting time)
LQstddev (standard deviation of waiting time)
LQtotal (total waiting time of all customers)
departures (number of departures)
fileout (outputs results to text file)

life: creating statistics objects.

fifo class ***lifo*** (LIFO queue with statistics – same as *fifo*, only *enqueue* modified in obvious way).

fifo class ***priorityqueue*** (priority queue with statistics – same as *fifo*, only *enqueue* modified in obvious way).

EXAMPLE SIMULATION

The following is a simulation model of a single queue multichannel system with all usual statistics collected. The model is complete except the procedures that read the data and save the results. Active customer approach is used, comments make the text (hopefully) self-explaining. Parameters of *histotally* objects define the range and the number of bins. So far the parameters have to be adjusted manually according to preliminary results.

```
external class quesim;
quesim  begin
 real duration;       ! experiment length;
 ref(fifo) queue;     ! the queue;
 integer servers;     ! number of servers;
 integer busyservers; ! working servers;
 integer seedg, seeds;
      ! seeds of random generators;
 real mininter, maxinter;
      ! uniform interval between arrivals;
 real smean, sstd;
      ! normal service duration;
 ref(histotally) timespent,queuetime;
  ! statistics on time in system and queue;
 ref(accumulator) working;
      ! statistics on working servers;

 process class generator;
 ! generates repeatedly customers;
 begin
  while true do begin
    activate new customer(0,2);
     ! parameters: priority and unit cost;
    hold(uniform(mininter,maxinter,seedg));
     ! uniform intervals;
  end while;
 end of generator;

 g_customer class customer;
     ! only life is specialized:;
 begin
  ref(customer) next;
  if (busyservers = servers) then
    waitin(queue)
      ! has to wait if all servers busy;
  else
```

```
    nowaitin(queue);
      ! needed for queue statistics;
  queuetime.update(time - timeinq);
      ! updating waiting time statistics;
  busyservers := busyservers + 1;
      ! service starts, seize a server;
  working.updateto(busyservers);
      ! updating working servers;
  hold(normal(smean, sstd, seeds));
      ! this is the normal service;
  busyservers := busyservers - 1;
      ! release the server;
  working.updateto(busyservers);
      ! updating working servers;
  if not queue.empty then begin
      ! first from the queue served:;
    next :- queue.takefirst;
    activate next after current;
  end;
  timespent.update(systemtime);
      ! systemtime is g_customer's method;
 end of customer;

 procedure readdata;  begin ... end;
      ! reading data from a text file;
 procedure writedata;  begin ... end;
      ! saving results to a text file;

 readdata;      ! main program body;
 busyservers := 0;
 queue :- new fifo("Q1",maxint);
              ! can have limited capacity;
 timespent :- new histotally(0,16,15);
 queuetime :- new histotally(0,10,15);
 working :- new accumulator(0);
 activate new generator;
 hold(duration);
 outtext("Run duration [s]: ");
 outfix(cputime,3,8); outimage;
 writedata;     ! evaluation;
end of program;
```

The results contain the distribution of the time spent in the system – see the above example and the distribution of the time spent in the queue. The following extract from the results file shows the statistics on the queue and on the servers (there were 5 service channels).

```
The Queue queue statistics:
Average length 0.09
Length Std Dev 0.35
Maximum length 5
Current length 0
Average waiting      0.19
Av nonzero wait      0.22
Maximum waiting      7.84
Waiting Std Dev      0.59
Total waiting        917.82
Total wait cost      1835.64
Arrivals             4959
Not waiting arr      4143
Departures           4959
Total busy time      763.45
Utilization [%]      7.63

Number of busy servers statistics:
Average         3.97
Std Dev         0.86
Minimum         0.00
Maximum         5.00
Utilization of servers:  0.794
```

CONCLUSION

There is a choice how to build a discrete simulation model,. Either use an interactive GUI environment like Extend™ or Arena™ or program the model in a discrete simulation language or a tool. Drawing models by using a mouse is comfortable for education and simple models, but complex models especially those with dynamically changing topology have to be programmed. SIMULA is one of many existing tools. It's use needs some initial training. My experience is that IT students that know another OOL like Java are able to write medium-scale simulation models after about 5 hours of introductory lectures. The reward is a language with clear lucid constructs. Mechanism of main classes enables easy expansion and re-use of the code by using the object model itself. Now there is a free PC version of SIMULA, the classes described in this paper will also be freely available – for both visit the author's page. Alternatively there is DEMOS with more advanced simulation support. In near future the class *quesim* will be expanded by some more useful local classes like multichannel servers and whole service stations.

REFERENCES

Birtwistle, G.M.; O.J. Dahl; B. Myhrhaug; and K. Nygaard. 1973. *SIMULA Begin.* Auerbach Publishers.

Birtwistle, G.M. 1979. *A System for Discrete Event Modelling on SIMULA (Demos)*. Macmillan Press.

Bratley, P.; B.L. Fox; and L.E. Schrage. 1987. *A Guide to Simulation – 2nd Edition*, Springer.

Eckel, B. 1998. *Thinking in Java.* Prentice Hall.

Extend User's Manual. 1997. Imagine That Inc., San Jose.

Holmevik, J.R. 1994. "Compiling SIMULA: a historical study of technological genesis". In *IEEE Annals of the History of Computing,* Vol 16, No 4, 25-37.

Kelton, W.D.; R.P. Sadowski; and D.A. Sadowski. 2002. *Simulation with Arena*, McGraw Hill.

Kirkerud, B. 1989. *Object-Oriented Programming with SIMULA*, Addison-Wesley.

McNeley, J.L. 1968. "Compound Declarations". In *Proceedings of IFIP Working Conference on Simulation Languages*, Oslo, May 1967. North Holland, 292-303.

Pooley, R.J. 1987. *An Introduction to Programming in SIMULA*, Oxford, Blackwell Scientific Publications.

Russell, E.C. 1983. *Building Simulation Models with Simscript II.5*, CACI Inc., Los Angeles.

SIMULA Standard. 1986. Simula a.s., Oslo.

Sklenar, J. 1997. *INTRODUCTION TO OOP IN SIMULA.* Available at http://staff.um.edu.mt/jskl1/talk.html.

SIMULATION OF FMS INCLUDING AUTOMATED GUIDED VEHICLE

Eugene Kindler
Ivan Křivý
Ostrava University Faculty of Sciences
CZ – 701 03 Ostrava, Dvorakova 7, Czech Republic
E-mail: kindler@ksi.mff.cuni.cz, ivan.krivy@osu.cz

Philippe Lacomme
Alain Tanguy
LIMOS, CNRS UMR6158, Université Blaise Pascal,
Complexe des Cézeaux, 63177 Aubière Cedex, France
E-mail: {lacomme, tanguy}@isima.fr

KEYWORDS
Discrete Event Simulation, Reflective Simulation, SIMULA, Flexible Manufacturing System, Job Scheduling

ABSTRACT

This paper focuses on the improvement of simulation models for the development of a simulation based scheduling methodology. This methodology was designed for an AGV-based automated manufacturing system by simultaneously dealing with material processing and transportation functions. This approach is based on an iterative procedure: a stochastic algorithm determines a job input sequence and a deterministic simulation is performed in order to investigate the system performance and the makespan, then the stochastic algorithm improves the initial job input sequence using evaluated makespan. During the system functioning a deadlock may arise. In order to avoid such a situation we may reject the job input sequence but we prefer the tuning of the job input date using reflective simulation. So, we control the consequences of a new job input by means of an internal simulation inside the main one, and the job and vehicle schedules are validated dynamically using given vehicle and machine dispatching rules. Such an iterative method requires efficient and well-implemented simulation models. We tested C++ and Simula for new simulation model implementation, we propose a reflective simulation model, an improvement approach to increase simulation speed and we discuss some obtained results.

INTRODUCTION

Flexible Manufacturing Systems (FMS) are designed and implemented to gain flexibility and efficiency of production. They are equipped with several machines and Automated Material Handling System (AMHS). So as to realize a FMS with full potential for flexibility, we must exercise careful planning and control of the design and management of its AMHS. One of the most difficult operational problems in FMS is the suitable management of the production sequencing and the allocation in time of required resources (Blazevich 1994).

The analysis of 50 existing FMS proposed in (Kusiak 1988) shows that the most commonly used material handling devices are: material handling robots, gantry robots and automated guided vehicles (AGV). We also regard the following handling devices: rail cars for shuttle and carriers for monorail systems. The systems using these types of handling devices are trip-based automated material handling systems. The trip-based AMHS effectiveness depends on several factors, among them a well-designed device management whose primary functions (Tanchoco 1991) are:

- dispatching, process selecting and assigning tasks to material handling devices,
- routing, process selecting specific paths covered by material handling devices to reach their destinations,
- scheduling, process determining arrival and departure times of material handling devices along their prescribed paths so as to ensure collision free moves.

The joint use of mathematical programming techniques or heuristics and simulation has considerable potential to improve the state of knowledge in this area. As stressed in (Ganesharajah 1998), since the simultaneous scheduling of jobs and devices is a NP-hard problem it has been solved in using heuristic approaches, and because there are many locally optimal solutions in such problems, it would be useful to explore metaheuristic approaches. From our point of view, the best approach consists in generating a schedule in using mathematical modelling or heuristic optimization approaches and then testing the result by simulation to evaluate the performance criteria and feasibility of schedules under more realistic conditions.

We propose a simulation based scheduling methodology for approximately solving the joint job input sequencing and device dispatching problems. The modelling approach includes optimization, computer simulations, performance criteria analysis, and takes into account the impact on the problem solving of the following constraints: the limited input/output buffer capacity, the limitations on the jobs simultaneously allowed, the dynamic behaviour of the system and the impact of the device and machine blocking.

Modelling experiments of modern FMS allow us to regard AGV devices able to elaborate decisions automatically applying FMS management rules, and to simulate the consequences of their decisions for a short time horizon. For instance: an AGV should carry a new job into the system but this decision might cause a deadlock. It is not suitable to simply reject a job input sequence when it is possible to carry the job later. Furthermore, many experiments show us that it is very difficult to run a simulation backward when a system collapse arises. Therefore, we have to start a new simulation when a decision must be tested. Thus, the new simulation must be

nested into the main one. Moreover, the modern FMS may include computers having the ability to simulate. These facts cause some modelling and implementation problems coupled with the wishes of a simulationist who disagrees in writing almost the same model two times, but they were stimuli to define the reflective simulation.

PROBLEM STATEMENT

Trip-Based Automated Material Handling Systems

A trip-based AMHS consists of one or several self-powered handling devices that are able to operate independently in an asynchronous way. Many AMHS used in industry are trip-based handling systems, for instance: rail guided device with linear track, mobile material handling robots, gantry robots, automated guided vehicle systems (AGVS) and monorail systems. Each load unit moves in the system and it is served one at a time by only one of the devices. The rates of loaded travels between the pairs of pick up and drop off stations can be obtained from manufacturing process plans (operation sequences) or from a production schedule known during a certain period of time. Two types of production schedules can be considered:
- the job sequencing, which is the decision of determining the order in which the jobs enter in the manufacturing system and pass through the machines;
- the job scheduling, which is the decision of determining the job sequencing over the time domain.

Problem formulation

Since the number of jobs in the system is limited and the total processing time is constant (predetermined upon arrival of job), the only way to minimize the makespan is to reduce the waiting times due to the blocking and non availability of required resources. Therefore the objective is to find an order in which the jobs enter in the manufacturing system, and it best satisfies the following set of objectives: to reduce the cumulative lead time of the production schedule, to improve the machine utilization, to reduce the total device loaded and empty travels, to assure the system operation without deadlock.

This research deals with the combined problem of scheduling problem, n/m/G/Cmax (Conway 1967) which is a well known NP-hard problem (Lenstra 1978), and of generic vehicle scheduling problem (VSP) which is also a NP-hard problem (Orloff 1976). A literature review of VSP can be found in (Kusiak 1985). The addressed problem falls into the classification (Garey 1979) of NP-complete problems for which an efficient optimal procedure does not exist (Ganesharajah 1998). Based on its structure, the problem can be broken down into two subproblems and solved by an iterative procedure that tries to accommodate the combinatorial nature in finding the solution.

SIMULATION BASED SCHEDULING

The proposed methodology is embedded in an iterative search procedure. The procedure begins with the initial data for the problem. An upper bound (UB) is updated each time the model reaches a better solution. Initially, UB can be the makespan of a known solution (job input sequence) to the original problem, it is supplied by industrial users or it is found by a heuristics rule. The procedure stops when conditions are satisfied: maximum CPU time, number of iterations or user stop requirements…Taking into account the combinatorial complexity of the joint job input sequencing and device dispatching problem, we choose the Kangaroo (Fleury 1993) stochastic algorithm, which stems from the simulated annealing metaheuristic.

The purpose of this study is to analyse the FMS performance as well as makespan by taking into account not only the limited capacities of the machines but also the AMHS and the in-process inventory. Therefore, the evaluation model must be developed in such a way that these resources and their interactions are represented in details. By this way simulation is used to evaluate the FMS performance criteria for an application in relation to the dispatching rules in force.

Performance Criteria

Performance criteria, which can be monitored in a simulation, are either AMHS-centred or FMS-centred. The ones referring to FMS are work station utilization, FMS output, makespan and station buffer levels. The AMHS related are their utilization, idle and blocking time. For trip-based AMHS like AGVS are also very useful the track utilization, block time for vehicles on the track segments and crossings and the time AGV travel empty to pick up a load.

The AMHS performance is evaluated by measuring the operating parameters of AMHS. An AMHS requires an amount of time to transport the whole volume of load units. It is spent for the following activities: travelling empty to a load station, loading, travelling full to an unload station, unloading, loaded and empty blocked (wait for loading, unloading at stations and for passing through segments, crossings and switches), idle stopped and travelling idle (wait for dispatching). In this context the required criteria to evaluate the transport efficiency are the average times spent by AMHS devices in each of the above-defined states during the studied period.

A Special Purpose Discrete Event Simulator

In order to satisfy the time constraints of the iterative framework a special purpose package using PASCAL programming language, SIMAMHS (Gourgand 2000), has been first developed for the simulation of trip-based AMHS based FMSs to extract the dynamic behaviour of the entire FMS. Based on a generic object-oriented model of the described FMS class, SIMAMHS has a flexible design and can handle an arbitrary user-specified layout for an FMS. By collecting and summarizing essential statistics on system performance the package provides a complete bottleneck analysis as well as the needed performance criteria evaluation at the end of each simulation experiment.

This simulator provides significant results (Gourgand 2000). However, a highly modular tool is required to best satisfy several aims. We have to explore FMS modelling with more than one vehicle so new model of traffic, routing and more generally complex management rules must be evaluated and validated. A simulation trace analysis and an animation tool help us to solve this first problem. The system behaviour study imposes computing of more or less performance criteria on developers. Moreover, modellers and simulationists use intensively object-oriented tools, complex data structures and suitable graphical user interfaces. These constraints and facilities have a CPU time cost that is incompatible with a fast iterative method based on about 20000 simulation runs.

REFLECTIVE SIMULATION

The classic simulation is commonly applied during the design of modern systems. Some of them include one or more computers and thus the ability to simulate their future behaviour. Other systems include intelligent components, such as drivers, who are able to determine their shortest path among obstacles and to elaborate decisions. Everybody knows that a simulation run is time consuming and other methods are better but they demand more time for their implementation, their tuning and their adaptation when the system in design evolves. Beside its worst aspect, the simulation is often the most suitable method to take into account many modifications of the system structure or new management rules and to validate decisions taken during the system functioning. Thus, we have to regard the simulation of simulating components and to study the nesting of an internal model into an external one.

Reflective Simulation Implementation

The implementation of reflective simulation taught that it is a difficult activity because of psychological aspects of the modeller and of a risk of mixing elements of both models. Therefore, we had to define the reflective simulation, the nested models, their nesting level, a methodology to design reflective simulation models (Kindler et al. 2001, 2002a, 2002b; Křivý et al. 2002). It appeared that we need a simulation language being object-oriented, block-oriented and able to manipulate local classes, life rules and quasi-parallel systems (QPS). Only four candidates were found: Beta, Java, MODSIM and Simula. Limitations concerning QPS eliminated Beta and those concerning local classes eliminated MODSIM. Java is not secure regarding element mixing of the different models. Thus, only Simula is able to implement reflective simulation in a secure way. Mainly, there are three free Simula compilers: CIM, Lund and Simula a.s. compilers covering the user's needs under UNIX, Linux and Windows.

Reflective Model Design

In order to model manufacturing systems using reflective simulation we started to convert the QNOP library (Tanguy 1993; Gourgand and Tanguy 1993). It includes queuing network objects, management rules, statistics facilities and predefined manufacturing objects managed by implemented rules. The library is open and it is easy to add new objects and rules. We discovered that the translation could be done by software. This way is more interesting to spare the modeller's efforts and to include more control in the reflective simulation model design. The process could by applied to other libraries (external classes) so as to obtain a new library for reflective simulation of different systems.

EXPERIMENTS ON SIMULATION MODELS

We are used to consider two kinds of models: the knowledge model and action models. The knowledge model describes a generic and unified representation of the system class. It results from an object-oriented analysis and contains the specification of system entities and their functioning. An action model describes one system instance; it is obtained in extracting and instantiating entities and rules of functioning from the knowledge model.

In order to obtain an efficient simulation model we have to prune the initial model and to modify data implementation. Two languages are tested: Simula and C (and its variants). In addition, two approaches are applied: event approach in C and process approach in Simula. This first step of simulation model implementation presently considers only one vehicle in the FMS and aims to obtain fast simulation models. We first describe the system under study.

Networks and Data

For testing, we choose four different layouts (Fig. 1) presented in (Kusiak 1985) and five job sets, which are identical to those presented by (Ulusoy 1993). For each layout there are four FMU and one Input/Output station. Each FMU has an input buffer and an output buffer, both with limited capacity. The buffer capacity is limited to two places.

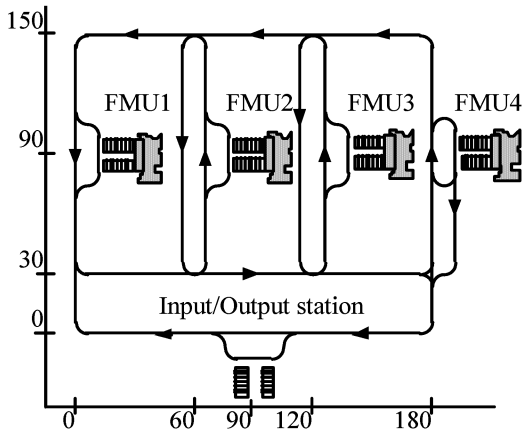


Figure 1: The First FMS Layout

The Input/Output station is assumed to have a sufficient capacity to store all jobs to be scheduled. The other characteristics of these systems are:
- the pick up and drop off operations at stations are assumed to take 1 minute,
- the vehicle travelling speed is constant: 0,5 m/s,

- the initial location vehicle is at FMU3,
- acceleration and deceleration have not been considered,
- any battery recharging is done during the off shift period,
- each load station and unload station can be reached by AGV located at any other station,
- vehicle breakdowns are negligible,
- bypasses are incorporated only in front of each pick up/drop off station.

The machine dispatching rules are used to select the job from the input buffer upon the completion of the previous operation. The machine dispatching rule is FIFO. We consider the following device dispatching rules:

- Idle device: the device stays idle at its last location and waits for request.
- FCFS (First Come First Served rule): the device is dispatched to the earliest job completion date.
- STT/D (Shortest Travel Time/Distance rule): this rule minimizes the travel time of empty devices. The device is dispatched to the nearest FMU.
- MQS (Maximum Queue Size rule): a device is dispatched to the FMU whose output buffer has the largest number of jobs. If two or more FMU have the same output buffer size then the conflict is solved using the redundant device dispatching rule STT/D.

The number of jobs simultaneously allowed in the FMS is a function of the total number of processing positions in the FMS. Five congestion levels are defined as 1, 1.5, 2, 2.5 and 3 times the total number of processing positions.

C++ Simulator

We implemented an event-oriented simulator based on a slightly modified initial model (Fig. 2) using the C++ language, and realized a graphical character animation of simulation trace in order to validate the simulator. The simulator accepts a specification file of a system and provides a result file and a formatted trace file. The trace is inhibited to gain in speed. The simulator was developed as a large mainframe application (29 module files) with Visual C++ under Windows. Severe compatibility problems arose: 9 compilers won't compile the programme and in spite of reuse chart respect, the simulator is not easy to modify.

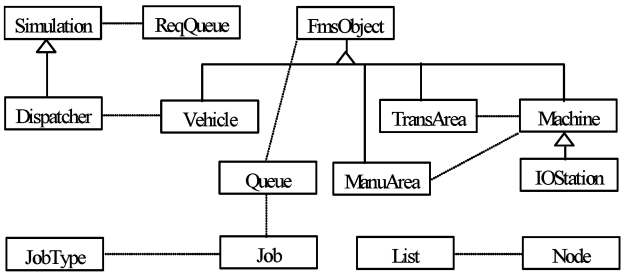


Figure 2: Class and Simplified Relationship Diagram

First Simula Simulator

The first process-oriented simulator is built in Simula language and it is composed of an external class of FMS objects (200 lines) and a main programme that reads a system specification file. It is very easy to automatically generate a Simula programme from the system parameters. The basic parameters are passed as external class parameters, the procedure *GenDist* generates the distance matrix (FMU to FMU), *GenJT* generates the job types and *GenJob* creates the jobs. The external class *FMSAMHS* contains two process classes: *CVehicle* and *CMachine*, one link class *CJob* and needed data structures. The *CVehicle* life rule (coroutine) includes the dispatching rules. For instance, the method *FindMove* (Fig. 3) searches the nearest FMU (machine) having the least number. We can see synchronisation procedures: *hold(0)*; *reactivate current after nextev*; *reactivate main*. The life rule of the external class contains an instruction *inner* so it runs into three phases: initialization, prefixed block execution (system simulation) and simulation report printing. This report is more or less detailed depending on the running model.

```
procedure FindMove;
    begin
FMBeg:  hold(0);                                              ! may be last event at this time;
      if Fmu(Pos).outjob > 0 then Dest:=Pos                   ! don't look elsewhere;
      else  begin integer destmin; real distmin;
        destmin:=maxint; distmin:=maxreal; Dest:=NbFMU;
Search: if fmu(dest).outjob <= 0 then goto OneOther;          ! search nearest destination;
        if dist(Pos,Dest)<=distmin then begin destmin:=Dest; distmin:=dist(Pos,Dest); end;
OneOther:  Dest:=Dest-1; if Dest >=0 then goto Search;
        if destmin < maxint then Dest:= destmin               ! found destination;
        else  begin                                           ! look for the next event;
          if NbJob <> NbProcessedJob then
          begin reactivate current after nextev;              ! wait for next event;
            IdleT:=IdleT+time-lasttime; lasttime:=time; goto FMBeg;    ! idle time period;
          end
          else  begin SimEnd:=true; Makespan:=time; reactivate main;   ! end of simulation;
          end; end; end;
    end;
```

Figure 3: Vehicle Method FindMove

In order to gain in speed we apply the following rules:

- simplify the object-oriented model structure,
- optimize algorithms, prune unnecessary output,
- simplify data structures,
- limit indirect access and reference depth if possible,
- reduce method and procedure call number,
- limit block generation number and depth,
- reduce garbage collection.

This improvement approach must be applied with a full validation of each new model, we may remove simulation trace and control output in the last version. In fact, it is a profile problem - and to a lesser extent a metric problem - for an object-oriented, block-oriented and coroutine-oriented language. In spite of difficulties, we increase

significantly the model execution speed. For a moderately optimized model and 1000 simulation runs the CPU time reached 6.1 s on a Pentium 200 MHz: it is about 10 times faster than the initial model. A modern Athlon 2600+ executes this task in less than one second.

Reflective Simulation Models

The external model design is relatively easy. The internal model is nested in the vehicle object (AGV). In the external model, the main loop of vehicle life rules performs the following tasks:
- The highest priority is given to the loading of a new job into the system. So, when the number of jobs in the system is less than the maximum number and the destination input buffer is not full, the vehicle moves to the input station and starts the internal model simulation including the new job input in the internal system. When a deadlock occurs the new job entry is delayed for a duration depending on the destination input buffer availability, else the new job is processed.
- In the system, the waiting jobs are processed according to the FCFS machine policy.
- The vehicle stays in place if no jobs need to be carried.

The internal model of a vehicle is slightly simple because new jobs are not allowed to enter the system. It signals a permission to enter by the modification of a vehicle attribute. The design of the internal model is not so simple because it has to generate and to initialize its objects as well as the present state of the sequencing set (SQS) for the internal simulation.

The execution time of the internal simulation represents a few percentage of the whole simulation time because the internal simulated time is spent to empty the system without a new job entry. Of course, it depends on the system load. It is possible to shortcut or to stop early the internal simulation when some conditions insure that no deadlock may appear.

CONCLUSION

We address the problem of integrated job input sequencing and device dispatching in a general job-shop with WIP consideration, and limited capacities of buffers and resources. An effective simulation based scheduling methodology has been developed for this purpose. It includes stochastic optimization and computer simulation supplying deadlock-free job and device schedules. Because the iterative method needs efficient simulation models we proposed two implementation tools. The Simula language appears to be more suitable than C++ because it includes advanced simulation primitives, has excellent legibility and enables developers to implement complex models in few lines. Furthermore, Simula belongs to the small set of languages able to implement reflective simulation models. Nevertheless, C++ could provide suitable results, but with more development, control and validation time. We proposed an approach for simulation model improvement. Experiments on a significant system shown it is possible to run 10000 to 70000 simulations in about less than one minute, depending on the computer and the model. We may expect an important issue in real-time context using reflective simulation. Further research could be directed to extend our approaches with other objective functions.

REFERENCES

Blazevich, J.; R.E. Burkad; G. Finke and G.J. Woeginger. 1994. "Vehicle Scheduling in Two-Cycle Flexible Manufacturing Systems" Mathematical Computer Modelling, 20 (2), 19-31.

Conway, R.W.; W.L. Maxwell and L.W. Miller. 1967. Theory of Scheduling., (MA: Addison-Wesley Reading).

Fleury, G. 1993. "Méthodes stochastiques et déterministes pour les problèmes NP-Difficiles", Doctorat d'informatique, Université Blaise Pascal, Clermont Ferrand II.

Ganesharajah, T.; N.G. Hall and C. Sriskandarajah. 1998. "Design and operational issues in AGV-served manufacturing systems" Annals of Operations Research, 76 , 109-154.

Garey, M.R. and D.S. Jonson. 1979. Computers and Intractability: A guide to the Theory of NP Completeness., (San Francisco, CA: Freeman).

Gourgand M. and A. Tanguy. 1993. "A SIMULA Object-Oriented Modelling Environment." In European Simulation Symposium 1993, October, Delft, The Netherlands, 688-693.

Gourgand, M.; P. Lacomme and N. Tchernev. 2000. "Simulation based scheduling with WIP consideration for FMS using Trip-based material handling systems" ESM2000, Ghent, Belgium, 481-488.

Kindler E.; I. Křivý and A. Tanguy. 2001. "Reflective simulation of discrete logistic and production systems". In: E. H. J. Kerkhoffs and M. Snorek (eds.): Modelling and Simulation, ESM2001, Prague, June 6-9, 2001. SCI, Delft, 861-865.

Kindler E.; I. Křivý and A. Tanguy. 2002a. "Towards Automatic Generating of Reflective Simulation Models". In: Proceedings of MOSIS'02, Vol. I, MARQ, Ostrava, Czech Republic, 13-19

Kindler E.; I. Křivý and A. Tanguy. 2002b. "Reflective simulation of logistic and production systems" In: A. G. Bruzzone, Z. Merkuryev, R. Mosca (editors): International Workshop on Harbour, Maritime and Multimodal Logistics Modelling & Simulation HMS, MAS 2002, Genoa, 97-102

Klein, C.M. and J. Kim. 1996. "AGV dispatching" International Journal of Production Research, 34 (1), 95-110.

Křivý, I.; E. Kindler and A. Tanguy. 2002. "Software for Simulation of Anticipatory Production Systems". International Journal of Computing Anticipatory Systems, Vol. 11, 320-335

Kusiak, A. 1985a. "Material Handling In Flexible Manufacturing Systems" Material Flow, (2), 79-95.

Kusiak, A. 1985b. "Flexible Manufacturing System: a Structural approach" International Journal of Production Research, 23 (6), 1057-1073.

Kusiak, A. 1988. Artificial Intelligence and CIM Systems, ed. A. Kusiak, Artificial Intelligence Implication For CIM, (Kempston : IFS (Publications) Ltd, Bedfort MK42 7BT, UK, and Springer-Verlag, New-York),1, 1, pp 3-23.

Lenstra, J.K. and A.H.G. Rinnooy Kan. 1978. "Complexity of Scheduling Under Precedence Constraints" Operations Research, 26 , 22-35.

Orloff, C.S. 1976. "Route Constrained Fleet Scheduling" Transportation Science, 10 (2), 149-168.

Tanchoco CO, C.G. and J.M.A. 1991. "A Review of Research on AGVS Vehicle Management" Engineering Costs and Production Economics, 21 (1), 35-42.

Tanguy A. 1993. "Implementation and Application of a Modelling Environment for Manufacturing Systems". In Proceedings of the 19th ASU Conference. Aberdeen, Scotland, sec. B-2.

Ulusoy, G. and U. Bilge. 1993. "Simultaneous Scheduling of Machines and Material Handling System in an FMS" International Journal of Production Research, 31 (12), 2857-2873.

TRAFFIC SIMULATION TOOLS

The "Simulation of Urban MObility" package: An open source traffic simulation

Daniel Krajzewicz, Markus Hartinger, Georg Hertkorn, Peter Mieth, Christian Rössel, Dr. Peter Wagner, Julia Ringel
German Aerospace Centre, Institute for Transportation Research
Rutherfordstr. 2
12489 Berlin
Germany
E-mail: Daniel.Krajzewicz@dlr.de, Markus.Hartinger@dlr.de, Georg.Hertkorn@dlr.de, Peter.Mieth@dlr.de, roessel@zpr.uni-koeln.de, Peter.Wagner@dlr.de, Julia.Ringel@dlr.de

KEYWORDS

traffic simulation, road traffic, open source, car-driver model, traffic research

ABSTRACT

SUMO is the acronym for "Simulation of Urban MObility" and is an open source project concerned with the development and usage of a traffic simulation. The project is a part of our scientific work concerned with the verification of different microscopic models of traffic, and their comparison ([1]). Further, the traffic science community often involves ideas where each of them needs a traffic simulation to be validated. Over the time, many more or less sophisticated simulations have been developed to do this job. They mostly stay unknown. This approach is not only very inefficient as a traffic simulation has many things to regard; also, the results are often not replicable or at least hard to compare. When a common platform is supplied, such problems should not occur.
Within this publication, we would like to introduce our package to the public in the hope to gain some further interest.

INTRODUCTION

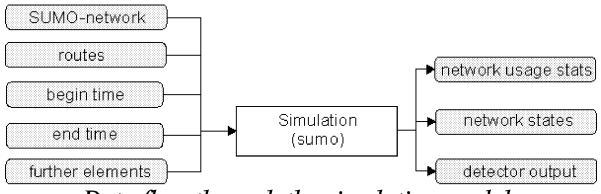
The development of SUMO began in 2001. With some experience in building simulation packages, namely FastLane ([4]), the main concepts were already known at the begin of the work. The simulation is based on the space-continuous, microscopic car-following model invented by S. Krauß. It is very well described and investigated in [8] and [5]. Other models that describe the movement of vehicles should be easy to implement. This is one of our main interests as one of our scientific topics deals with the comparison of traffic models regarding both their efficiency (execution time) and their ability to replicate the reality.
For a more detailed description of different models and model types, the reader is referred to [2] or to our internet pages, where you can find some models at the pages of the Clearing House for Transport Data and Transport Models [12] or some further SUMO-documentation describing it ([6] or [7]).
During the development two guidelines were followed. Firstly, the simulation has to be as fast as possible in execution. The current implementation is capable to simulate about 1 million of vehicles in real time on a 1GHz PC. The graphical user interface which uses the industry-standard library openGL [10] for visualisation is capable to deal with large city areas made up of several ten thousands streets on a normal PC without any problems and staying smooth. Further, the simulation shall be as portable as possible. Using the standard c++ language and open source or free libraries only, allows us to compile and run the package both under MS Windows and Linux and it should also be compilable when using SunOS or MacOS.

THE SUITE

The package consists of six main applications and some further helping programs. We will describe the main components.

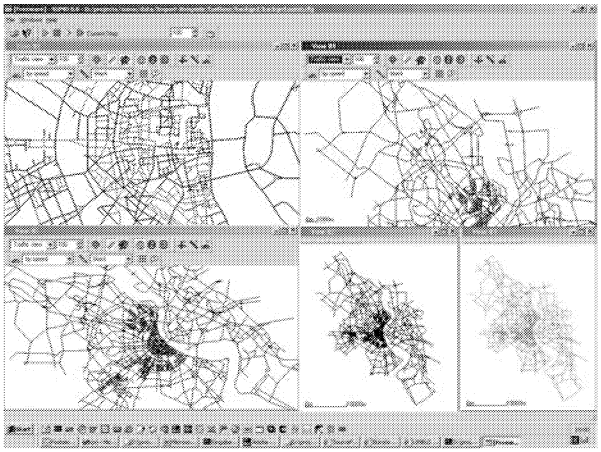
The simulation and its graphical version

The main application of the suite is of course the simulation itself. Two versions exist, a command line version without any visualisation, which is meant to be used during iterations when optimising networks without the need of interaction with a user. The graphical version is a little bit slower due to visualisation itself. As the graphical version is just an extension of the command line version, both applications use the same input and generate the same output.
Beside a network description, which has to be converted into a proper format using the net-conversion application, the routes the vehicles use and the vehicles themselves must be supplied to the simulation. As output, the simulation may generate aggregated street values such as the flow, the mean speed or the usage rate, discrete information about vehicles' positions and speeds, or simulated detectors such as induct loops known from real life or some more sophisticated detectors that cover longer surfaces of a street's part.



Data flow through the simulation module

The simulation handles multi-lane traffic with lane changing. No vehicle collisions occur under normal circumstances.



A screenshot from a simulation of the city of Cologne, the Centre of Applied Information Science is located at, showing different zooming scales and coloring schemes (down right: streets colored by the maximum speed allowed)

The network generator

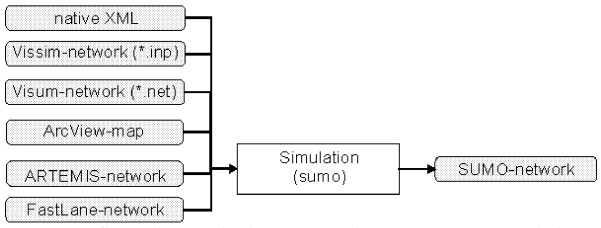
When dealing with large urban areas, no one likes to implement them street by street using an editor - beside the roads themselves, connections between them have to be specified and information about the traffic lights and right-of-way rules have to be set. To avoid such manual editing, our approach is to import digital maps and to automatically generate all information needed to describe the network.

This process is quite complicated, involving about ten steps for the computation of dependencies between the roads only. Further, right-of-way rules and traffic lights have to be computed. Still, the building of large cities such as Munich's network takes less than five minutes on a normal PC. Doing the same by hand would need several weeks. Surely, the heuristic we use do not cover all possible ideas road designers may have. They will be validated in future, but do they prove their quality during our current projects already.

Currently the network converter handles the following file types:

- Maps stored in the Arc View-format
- Networks from FastLane
- Networks from Artemis (in work)
- Networks from Visum (version 7.0 up to version 8.0)
- Networks from Vissim (version 3.6 up to version 3.7)
- An easy to generate native-XML-format

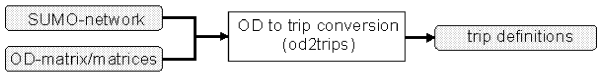
Different types of junctions are supported and computed: priority junction, right-before-left junctions and traffic light junctions. Traffic lights may both have a fixed switch plan or be controlled using different, simulated detectors.



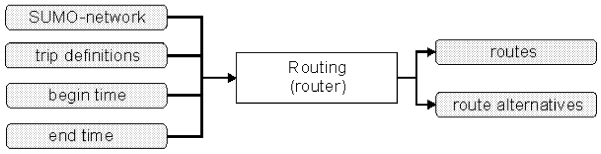
Data flow through the network conversion module

The traffic assignment - OD-matrices and routes

Detailed descriptions of a cities' mobility patterns are at least as needed for valid simulations as a detailed road network description is. To fill the network with traffic in SUMO, one may either use existing OD-matrices and convert them into route descriptions using the od2trips-tool or specify the routes themselves by hand. The first approach is the one used mostly within the traffic science when dealing with large real-world scenarios as such matrices are often available. The second one is needful when one is interested in agent-based simulations where each agent has its own wishes about his mobility.



Data flow through the od-to-trips converter



Data flow through the routing module

The loaded routes, being mostly described by their starting point and their ending point, have to be laid into the network. When all people are using the shortest path, this road will get jammed and other roads will stay free. To gain a realistic behaviour of the flow, SUMO uses the Dynamic User Equilibrium approach developed by Christian Gawron in his PhD thesis [4]. It assumes that the driver tries more promising (faster) alternative routes with decreasing probability over iterated simulations. Although a simulation has to be performed several times to achieve it, this approach yields in a realistic behaviour of the flow.

CURRENT USAGE WITHIN THE IVF

SUMO is already being used within several projects at the Institute of Transport Research (IVF) at the German Aerospace Centre where it was developed with the participation of the Centre for Applied Information Science.

To name only the important ones, it is now used to simulate the benefits of the optical information sensors (OIS) developed at the IVF. This is done by simulating a small area around our institute using information about the traffic amount gained by a measurement

campaign this year. After the verification of the simulation, the traffic lights currently used which do have a fixed time interval, will be replaced within the simulation by traffic lights with a new intelligence based on the information that may be gained by the OIS. Both, the OIS detectors and the traffic lights' intelligence will be implemented in SUMO and used to predict whether and how the traffic may be positively influenced – mainly reducing the waiting time on the traffic lights – when OIS systems are used. Final results will be available at the late summer of this year (2003). Within the second step, the network will be enlarged and simulated again to gain further information about the possible benefits.
Within another project, INVENT, we use the simulation to validate traffic management approaches. Herein, to gain the most exact information about the networks to be considered, we work together with the german traffic engineering company "ptv". INVENT itself is a large project leaded by most of Germany's road vehicle manufacturers and other companies concerned with traffic. The traffic management approaches we deal by now with, are mostly based on the optimisation of traffic lights. Later this year, other approaches from telecommunication networks, physics or biology shall be tested, too.

FURTHER PARTICIPANTS

By now, a few groups work on the package. It is used at the IFS, another institute for traffic research at the German Aerospace Centre for some projects by now and gains increasing interest there. Also, some interest comes from the University of Laval, Canada where it shall be a part of a multi-agent system including own software and another traffic simulation package, Paramics.

FUTURE WORK

As the software is stable by now and does contain everything needed to perform professional simulations of road traffic, the main work in the future will concentrate on the implementation of some more sophisticated data processing and visualisation. Further, some methods must be rechecked, such as the computation of junction shapes.
To achieve one of the major goals, the possibility to integrate different car movement models, a programmer interface for this part will be implemented.

CONCLUSIONS

With SUMO, we do offer a well designed, complex and modern simulation package for free, which does cover all steps needed to perform own simulations of large areas for free. As described above, we do already use it in some industry-near projects and it should be surely interesting for industry due to its extensibility and portability.
By doing this, we hope to gain some participation which allows to increase the software's abilities and usability. Beside this, we hope SUMO to be a shareable standard platform for models of any kind of algorithms needed within traffic simulations to make them more comparable.
You may find the projects web sites at:
http://sumo.sourceforge.net

REFERENCES

1. E. Brockfeld, R. Kühne, A. Skabardonis, P. Wagner. 2002. „Towards a Benchmarking of microscopic Traffic Flow Models". TRB, in submission
2. D. Chowdhury, L. Santen, A. Schadschneider. 2000. "Statistical Physics of Vehicular Traffic and Some Related Systems" Physics Reports 329 (4-6) (2000) pp. 199-329, Elsevier, 2000, ISSN: 0370-1573
3. Ellemtel. 1990-1992. Coding Standards Page. http://membres.lycos.fr/pierret/cpp2.htm
4. Christian Gawron. 1998. „Simulation-Based Traffic Assignment". Inaugural Dissertation.
5. Stefan Janz. 1998. „Mikroskopische Minimalmodelle des Straßenverkehrs". Diploma Thesis.
6. "SUMO (Simulation of Urban MObility); An open-Source Traffic Simulation". D. Krajzewicz, G. Hertkorn, C. Rössel, P. Wagner. In: Proceedings of the 4th Middle East Symposium on Simulation and Modelling (MESM2002), Edited by: A.~Al-Akaidi, pp. 183 - 187, SCS European Publishing House, ISBN 90-77039-09-0
7. "An Example of Microscopic Car Models Validation using the Open Source Traffic Simulation SUMO"; D. Krajzewicz, G. Hertkorn, C. Rössel, P. Wagner; In: Proceedings of Simulation in Industry, 14th European Simulation Symposium, 2002, Edited by: A. Verbraeck and W. Krug, pp. 318-322, SCS European Publishing House, ISBN 3-936150-21-4
8. Stefan Krauß. 1998. „Microscopic Modelling of Traffic Flow: Investigation of Collision Free Vehicle Dynamics". Hauptabteilung Mobilität und Systemtechnik des DLR Köln. ISSN 1434-8454
9. Los Alamos. 2002. „TRANSIMS: Transportation Analysis Simulation System"; C. L. Barrett, R. J. Beckman, K. P. Berkbigler, K. R. Bisset, B. W. Bush, K. Campbell, S. Eubank, K. M. Henson, J. M. Hurford, D. A. Kubicek, M. V. Marathe, P. R. Romero, J. P. Smith, L. L. Smith, P. L. Speckman, P. E. Stretz, G. L. Thayer, E. van Eeckhout, M. D. Williams; Los Alamos Laboratory; Vol. 3, 2002
10. OpenGL.org. 2002. OpenGL.org Homepage. http://www.opengl.org/
11. SUMO: G. Hertkorn, D. Krajzewicz, C. Rössel. 2002. SUMO Homepage. http://sumo.sourceforge.net
12. Clearing House for Transport Data and Transport Models website. 2003: A. Schulz, A. Parnitzke, P. Wagner. Traffic Cleainghouse Homepage http://www.clearingstelle-verkehr.de/ (german only)

An Approach for Dynamic Supply Chain Modelling

J. Manuel Feliz Teixeira
António E. S. Carvalho Brito

GEIN – Section of Management and Industrial Engineering
Faculty of Engineering of the University of Porto, PORTUGAL
Emails: feliz@fe.up.pt; acbrito@fe.up.pt

KEYWORDS: Supply Chain, model, simulation, operational policies, costs, inventory, transports, delivery.

ABSTRACT

This paper presents some concepts related with the development of a "model generator" for simulating supply chain systems. Such concepts arise not only from the object point of view taken over this system, but also from our intention to synthesise the behaviour of such kind of systems in a more general form, in particular concerning the events and activities related with *suppliers*, *factories*, *warehouses*, *retailers*, and even the last *customers*, which in fact trigger the flow of materials and information on the chain.

Unlike the static approaches usually used for strategic purposes, where parameters like the time of delivery or the average rate of material flow are held as inputs to the system, the ideas described here consider a dynamic representation of the supply chain in which those parameters are *results* of some detailed simulation process, what gives this approach the ability of modelling those systems starting from the less abstract point of view to a more abstract representation. The result is a more realistic picture of the dynamics involved.

Due to this fact, this approach could be seen as being more directed to managers than to strategists. Nevertheless, the objective is also that the models can be used to simulate either short or long periods of time, revealing their usefulness also for strategic analysis.

The basis of these ideas is to consider the flow of products, information and money between any elements in the chain as a general form of *customer-supplier* exchange activity, and also by treating each of those elements as inheriting from a single element which includes the basic behaviour (and resources) of a *factory*, *warehouse* and *retailer*. A description of such element will be made, explaining its structure and the associated *fixed* and *variable* costs of its various processes.

As we will see, many parameters considered inputs to other modelling techniques will appear here as outputs, giving the analyst more interesting data with which it can measure the supply chain performance by means of any statistical methods.

1. Introduction

During last decades, most of the simulation approaches used to model the supply chain have been deeply related with strategic intents, that is, were mainly devoted to study the problem of dimensioning and geographically positioning the chain components in a way the overall expected costs could be calculated and, maybe, optimised. The objectives of such approaches were mainly strategic, and so, dealing with average values over long periods of time. In a certain way, the intent of those models was to turn the results independent of *time* variable, and so, what one can retrieve from them is also a static expectation. Although these methods still are very important on the first step of a chain establishment on the ground, they cannot give confident results that could help to manage the supply chain, that is, they hardly would survive to the need of characterizing the chain dynamics, or optimising week-by-week the processes developing in it. To try to resolve this kind of issues, the supply chain started to be also modelled in a dynamic way, and the approach presented here must be considered one more perspective to enrich such trials.

In the point of view of this approach, the supply chain is already established on the ground and the problems to be studied or analysed must be related with the optimisation of the actual processes running in the chain. These processes are mainly the *purchasing*, the *stocking*, the *manufacturing*, the *delivering* and the *managing*; all of them will be measured using not only technical but also economical indexes. This last aspect will be included as a way to measure the economical performance of each element, being also very useful when the chain contains elements belonging to different enterprises, that is, when there is concurrence between different management systems implanted on the same ground. By simply representing the flow of money, one can visualise and easily calculate the *costs* and the *income* for a particular group of elements on the chain, and so, also for those one considers its own part of the system...

As showed in figure 1, in general the supply chain is considered a kind of network connecting *factories (F)*, *warehouses (W)* and *retailers (R)*, with these elements treated as having different structures and different behaviours on the processes running in the chain. In the *factory*, for example, there

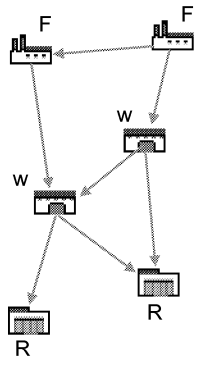


Fig. 1 General concept of a supply chain

is a transformation process that gives the row materials the aspect of products, while in a *warehouse* there is only storage and handling processes and in the *retailer* only the action of sales. Finally, what flow in the net are products, transported by vehicles.

Here, however, elements like *factories*, *warehouses* and *retailers* are considered inheriting from a single element that will be named *customer-supplier-unit* (CSU), with which will even be possible to model the figures of the *supplier* and *last customer*. Thus, the overall chain is seen as a group of CSUs connected together by a net of transports, where there also can be exchange of information (fig. 2). Now, what will distinguish a *factory* from a *warehouse* or from a *retailer*, or any other element, will only be how that particular element uses the resources of its CSU and the kind of materials it handles as input and output.

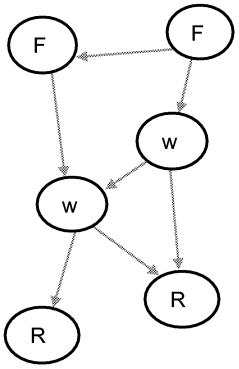


Fig. 2 CSU equivalent of a general supply chain

Each CSU lives from a constant exchange activity with its neighbours, receiving material and information and giving back money, delivering material and information and receiving money. Thus, in order to see how the economy develops between the elements on the chain, it will be included in the basic structure of the CSU a reference to its own *economic account*. One must not forget most of the decisions are taken not only based on the technical performance of the systems but also on how the money spreads along those systems... This account, of course, must include *costs* and *incomes*, in a way that with the results of the simulation one can better predict how each of these elements is contributing to the overall economical performance of the chain.

2. The *customer-supplier-unit* (CSU)

The CSU is then a basic element who owns a group of *customers*, to whom it delivers material and from whom it receives money, and a group of *suppliers*, from whom it receives material and to whom it returns money. Anyhow, to be able to handle this basic functionality, the CSU must also have in its structure some more general resources. Basically, these resources will be a *stock*, an *economic account*, an *input queue* where the *supplier* vehicles will wait to deliver the materials, and an *output queue* where other vehicles will line up to receive the material that must be delivered to the *customers*.

In order to be able to manage these resources, the CSU must also include in its structure a *list of suppliers*, a *list of customers*, a *list of products* and, finally, a *list of vehicles* representing its own *fleet*. In this approach we established as a principle that the *fleet* always belongs to the one who delivers, what seems a reasonable approach for most practical cases, even when the fleet is hired, as we will see later.

Considering this structure, which is depicted in figure 3, the CSU becomes a general element ready to be linked to other elements of its kind, making possible the overall supply chain representation. Of course, different CSUs can have different amounts of stock, different number of customers and (or) suppliers, different products, and so on, depending on the role they play in the chain. For instance, ultimate customers can be

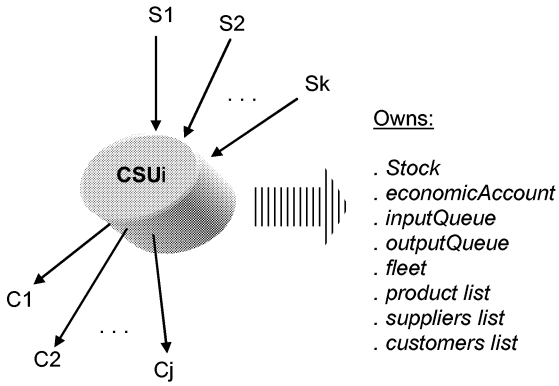


Fig. 3 Generalised CSU concept having *k* suppliers and *j* customers

thought as having null stocks, null fleets and null customer list, what will reduce them to a source of money for the chain and a demand of products. On the other end, a supplier source can be seen as a CSU having infinite stock and null supplier list. Note also that raw materials can be treated as products, even if later will be necessary to adapt the unity of transport to the proper case. Thus, this approach opens the representation also to *last customers* and *suppliers*, which in most of the simulators are treated as simple statistical parameters.

3. Connecting the CSUs and the flow of materials

To be able to model the chain dynamics is now necessary to connect each CSU to its effective neighbours. A net of *paths* in which vehicles will move ensures this. These *paths* can include normal roads, highways, railways, boat circuits, and even airlines. However, for didactic purposes, let us assume general-purpose ground vehicles as presented in figure 4.

Note that each *segment* of a *path* can then represent different costs to the delivery process; for instance, vehicles can first run on a highway and then change to normal roads. That, and the probable existence of real road junctions in practice, implies also the usage of the concept of *node* (fig. 4). Therefore, vehicles will travel from *node* to *node* with the objective of executing a certain predefined *job*. This also means each CSU must include the dimension of a *node*. Of course, as the vehicle moves from one *node* to the next *node* the delivery costs are expected to rise, as

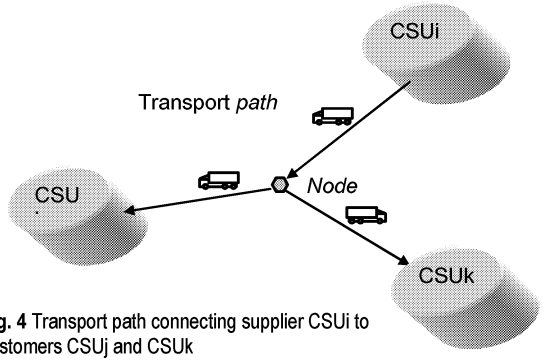


Fig. 4 Transport path connecting supplier CSUi to customers CSUj and CSUk

they are calculated based on the known costs of the vehicle utilization per kilometre and driver cost per day.

Note also that, in principle, time spent by the vehicles in these *paths* will not be predictable, also because different vehicles from different suppliers can reach the customer around the same time, and then it will be necessary to queue before the effective delivery will take place. Situations like this will be, of course, dependent on the layout of the chain, on the quantity of suppliers serving

each customer and on the intensity of delivering flow in the net, as well as on the delivering policy and the size of vehicles.

Each vehicle is treated as an independent entity, and then the movement of a vehicle trough these *paths* will be simply dependent on the vehicle itself, that is, it will result of the execution of a sequence of events that are "property" of the kind of vehicle. Thus, from a high-level point of view, there will be only two commands (events) each CSU can send to its vehicles: *e_startDelivery()* and *e_startReturn()*. The first sent after attributing a certain *job* to the vehicle and the later when a *job* is finished. The vehicle itself must then be capable of delivering the material without any other interference from its CSU, even if a *job* includes more than one delivery point, as in the case of local delivery.

4. Delivering speed

After the material is ordered to a particular CSU, it will start a sequence of actions on that CSU that will lead to the *processing of the order*, the *material packing* and finally the start of the *delivery process*. From the point of view of the customer, the speed of the delivery will also depend on the time spent on such sequence of actions and on the criteria used by the supplier CSU to assign *jobs* to vehicles. Over this, however, it must be added the time needed to carry the material along the *paths* between the CSUs, which is dependent not only on the *vehicle speed*, but also on the *limit speed* of each *segment* on the *path* trough where the vehicle must travel. The figure 5 helps to explain better these ideas, representing a *path* with 3 segments and 4 *nodes* connecting supplier CSUi to customer CSUj.

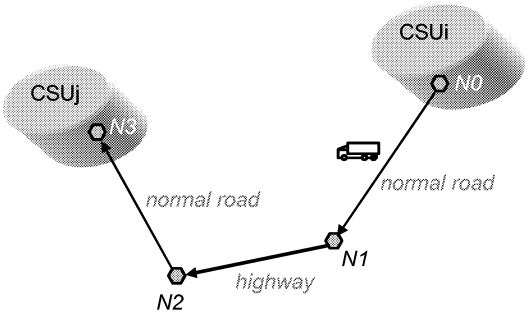


Fig. 5 Path connecting supplier CSUi to customer CSUj including normal roads and highway

If one assigns a *limit speed* to each *path segment* and a certain *vehicle speed* to each *vehicle*, then the effective speed of transport in that particular *segment* can be calculated as:

$$transportSpeed = Min(segmentSpeed, vehicleSpeed)$$

Thus, each time the *vehicle* reaches a new *node* leading to a new *segment*, it must recalculate its next speed of transport. As it is easily understandable, the *limit speed* assigned to a *segment* can serve to model not only the legal limit speed of that road but also the usual conditions of traffic flow on it.

In order to model other more specific aspects related with the delivery process, nodes will be of four different types: *normal*, if it is a node where the vehicles just change to a new path segment; *CSU*, when the node refers to a CSU; *transfer*, when the node is used to transfer material from one vehicle to another; and *pause*, when it represents a certain delay in the delivering process. This, of course, can also contribute to the mean speed of delivery observed between CSUs.

5. Costs

Once the supply chain is seen as a dynamic system, the costs associated with it must also result from a dynamic process, that is, they will be calculated and affected to the respective CSUs by means of certain event executions in time. Thus, it is important not only to know the amount of a particular cost but also the time when it must be affected to the CSU. This will also allow a continuous calculation of *costs* and *incomes* during the simulation process, for example, important economic indicators for the supply chain manager.

Based on the idea that each process in a CSU must have its costs, and considering as main processes the *purchasing*, the *stocking*, the *manufacturing*, the *delivering* and also the *management*, as depicted in the figure 6, it was decided to specify the costs following this same kind of taxonomy.

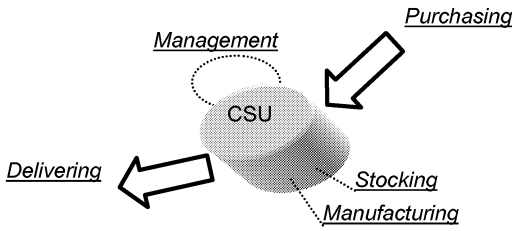


Fig. 6 Main processes for assigning costs to a CSU

We will talk about *purchasing costs*, *stocking costs*, *manufacturing costs*, *delivering costs* and *management costs*. Each of these costs can be thought as the result of other more specific costs, some of them assumed *fixed* and others *variable*, anyhow, it is important to bear in mind the more the costs are specified the more events one need to add to the model. Here we will only be interested on those costs usually considered more relevant, and each of them handled as a *fixed* and a *variable* component. The *fixed* will be considered the price to pay for "existing", while the *variable* will represent the price to pay for "operating" or "processing".

5.1 Purchasing costs

These costs must be affected to the CSU at the end of each *purchasing cycle*, that is, when products (or materials) are received from (and paid to) the supplier. The amount of these costs depends on the *price* previously established for the products by the supplier, as well as on the *administration* involved in the process. This last aspect is modelled as a cost per purchasing cycle, and contains the idea of the "*setup cost*" usually referred in *inventory management* literature. So, the purchasing cost will occur in the end of each *purchase cycle* and its amount will be given by:

$$PurchasingCost(t) = ProductsPrice(t) + administration(t)$$

Where *administration* can refer to invoice prepare, papers, phone-calls, etc., and also to personnel costs. Purchasing costs are considered here only as *variable* costs.

5.2 Stocking costs

The process of stocking includes many aspects, and so, many kinds of costs can be assigned to it. However, the most usually considered costs[1] are *operations*, which include receiving, order picking, replenishment, etc.; *administration*, including order processing, managing accounts, personnel, etc.; *occupancy*, which includes rents, services, insurance, depreciation, maintenance; and *inventory*, or *holding* costs, based on the existence of products in stock, usually 1% above the bank base lending rate. The overall stocking costs can then be expressed as:

$$StockingCost = [operations(t)+administration(t)\,] + \\ + occupancy + holding\ cost(t)$$

To turn the simulation process simpler, we will call *stock processing* cost to the *operations* and *administration* costs together, which are considered per unit of material handled. Then, the *occupancy* cost is based on a fixed cost per unit time (a fixed cost). And finally, the *holding* cost is based on the price of the money invested on the products in stock.

Stock processing costs and *holding* costs will be affected to the CSU at the end of each action of input or output on the stock (variable costs), while o*ccupation* costs will be affected continuously, as the time increases (fixed cost). It is important to notice that the *holding* costs begin when the CSU pays the product to the supplier and will only end when the CSU receives the payment for delivering it to the customer, thus also *in-transit* inventories are considered.

In general, the *holding* cost for the i^{th} product in the stock will exhibit the following dependence on time:

$$HoldingCost(t)i = (Cp)i \times (Np)i \times (HC)i \times (t\text{-}to)$$

Where:

$(Cp)i = i^{th}$ product unit cost.
$(Np)i$ = Quantity of products i in the stock.
$(HC)i$ = holding cost rate per unit time (bank lending rate)
to = Instant of time when the product was purchased
t = present time

There are also costs associated with *stock breaking*[2], each time the demand cannot be satisfied from the stock. These costs will be estimated by the quantity of material that have not been served, thus generating a *stockout*, as well as by the time spent to serve the customer compared with the time the customer accepts to wait for the materials to arrive, leading to a measurement of a "*customer satisfaction index*".

5.3 Manufacturing costs

In the present approach the factory is considered to be similar to the warehouse, with the differences that it will have two kinds (R and P) of products in stock (R for raw materials, P for finish products) and also a *process* named *manufacturing*, which will be responsible for the transformation of R products into P products. Basically, the manufactory is then seen as a warehouse having a frequent rearrangement of products in its inventory. Such a rearrangement is imposed by the *manufacturing* process.

As the figure 7 suggests, the costs associated with the process of *manufacturing* will mainly be the costs of *transforming* R products into P products (variable cost), the costs of *labour* (assumed as a fixed cost), and the costs of space *occupancy* (fixed cost) for the production zone. The costs associated with *stocking*

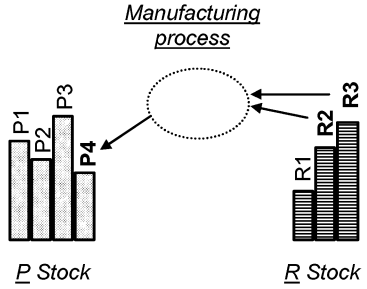


Fig. 7 Equivalent of a manufacturing process, in this case products *R2* and *R3* will give product *P4*.

will be the same referred in the previous section. The *transforming* costs must be specified for each product P in the factory, and given per unit of product. Of course, these costs must be affected to the CSU at the end of each *production cycle*.

Then, after a certain period of time t, the contribution of the *manufacturing* process for the overall *variable costs* of the CSU will be given by:

$$ManufacturingCost(t) = \sum [transforming(t)]_i$$

Where i referes to the *ith* product manufactured in that same period of time.

Finally, the *occupancy* costs related with the factory's zone, as well as the *labour* costs, will contribute to the global *fixed costs* of the CSU.

5.4 Delivering costs

Delivering costs are here considered as the result of two main contributions: *occupancy*, related with space reserved for the fleet, garages, or even rents, if the fleet doesn't belong to the CSU; and *transport*, which include the costs associated to drivers, vehicle consume of fuel and oils, paid roads, meals, delays, handling the materials, and so on.

Similarly to what was considered in the *StockingCost*, the *occupancy* cost is seen as a fixed cost, and given per unit time. The *transport* costs, however, will be computed during the simulation process each time a *vehicle* reaches a new *node* in the transport path network, and assigned to the respective CSU when the material reaches its destiny. Thus,

$$DeliveryCost = occupancy + transport(t)$$

The *transport* costs include the following main aspects: *fuel*, *lubricants*, *maintenance* and *roads* as costs depending on the distance travelled (ds); *drivers*, depending on the time travelled (dt); and *meals* and *delays*, as constant costs. As an example, if a vehicle is going from *node A* to *node B* with a speed v, the following *transport* costs will be added to its CSU when reached the *node B*:

$$Fuel(ds) = (VehicleFuelConsume/PricePerLitre)*ds$$
$$Roads(ds) = (PriceOfRoadPerKilometre)*ds$$
$$Driver(dt) = (DriverPricePerUnitTime)*(ds/v)$$

Where *lubricants* and *maintenance* can be considered included in the *fuel's* price. Notice that when the node corresponds to a node of resting or meal, also the constant costs of resting or meal must be added. Also notice that hiring the services of an external fleet can be modelled as keeping the cost of occupancy null.

5.5 Management costs

The *Management* costs, related with the costs of managers, analysts, secretaries, strategists, marketing, etc., will be considered here as *fixed* costs, and, as a first approach, they must be included in the fixed costs of the main resource of the CSU, that is, the inventory. These costs are also affected to the CSU as a continuously growing cost.

5.6 Total and global costs calculation

Each of the previous *variable costs* are recorded during the simulation as *instantaneous* values, thus, the calculation of their *totals* will automatically result from the simply sumation of those values along the time domain. For instance, considering the output transport costs the function $g(t_i)$, which would have the form of a data output record from the simulation, the *total variable costs of transport* would simply be given by:

$$total = \sum_i g(t_i)$$

This is a general method used in this modelling approach to record the behaviour of the interesting variables during the simulation run, which simplifies future calculations. For example, it will turn the calculation of *global costs* extremely simple, not only in a particular CSU but even in the entire chain network.

6. Information and the flow of money

Other things flowing in the chain are *information* and *money*, which in the present approach we consider associated with instantaneous operations. At the moment, the *information* will mainly be related with the communications between CSUs during material ordering, but we expect to include more about that in the near future, mainly the transference of the clients demand to the CSU's suppliers, in order to transfer the demand upstream on the chain, a procedure claimed to improve the partner's flexibility.

But let's now concentrate on the *money* flow, related with the payment to the supplier in the moment the material reaches its destiny. We make the following assumption: money always flow in the opposite direction of the material flow, that is, each time one receives materials one has to pay back its price to the supplier. It is based on this process of exchange the chain will maintain its activity along the time. The payment is here considered to be instantaneous.

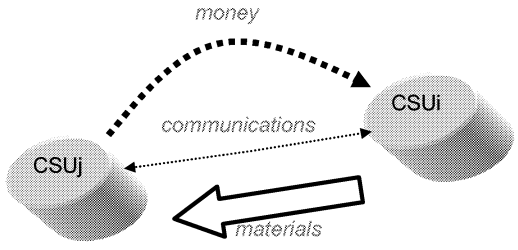


Fig. 8 Exchanging process between supplier CSUi and customer CSUj

The introduction of this economic aspect will give the analyst the possibility of studying the evolution of the chain parameters along the time not only based on technical performances but also taking into account economical aspects, as in fact it happens in reality.

Money flow will be triggered each time a customer receives material from a supplier. As the chain continues to work, such flow will spread to all the elements in the network, then allowing the instantaneous calculation of the *variable costs, incomes, total costs, etc.*, as well as certain important economical index, as the *cash flow*, for instance.

7. Modelling the general CSU activity

As we have seen, in each CSU some processes are running and interfacing with each other to create the complex activity of the unit. Those processes are the *purchasing*, the *stocking*, the *production*, the *delivery* and the *management*. Except the *management*, which is here considered instantaneous and coincident with any events in the CSU, each of the other processes contribute to the overall CSU's state with a certain number of states (we call *live states* those states which duration can previously be known or computed, and *dead states* those states where this cannot be done, as in waiting states and queues, for instance)[3]. The collection of such states, and the possible connections between them, will form the *state diagram* of the CSU, the real basis for the model. Therefore, first we have specified the *state diagram* of the CSU, and then converted it into a sequence of events which finally have been translated to a programming language code, in our case the C++.

Notice that, while building the supply chain network in the simulator, the user will no longer need to think on those internal states of the CSU, as they become endogenous to the CSU object, turning the supply chain representation more likely a game of objects connected between each other by paths of information and material flow. Thus, at the user's point of view, the paradigm used in the simulation is a *real object paradigm* and not any of the usual basic paradigms described in simulation literature like Petri-nets, activities, events, processes, 3phase, etc.

7.1 The purchasing process

This process is the process of ordering material to suppliers, and starts in each of the conditions: (1) The present stock control policy determines it is time to reorder. (2) There is an order from a client that cannot be fulfilled direct from stock and, in the case of accepting *backorders*, new material can be ordered.

After creating and preparing the new order with the material requirements, each of these "procedures" will then trigger the start of the purchasing process, which in fact reduces to two "events": (1) The sending off the new order to the respective supplier, we named *e_order()*. (2) The receiving of the material arrived, we named *e_arrive()*. Between theses events the process will be in a *dead state*, waiting the material to arrive from the supplier.

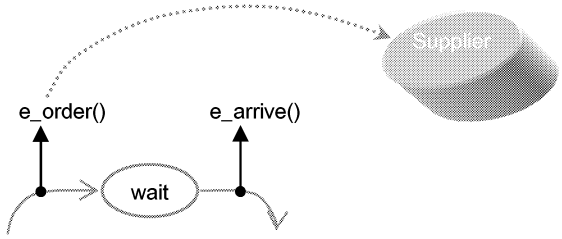


Fig. 9 The basic purchase process.

Notice that *lead times* are initially unknown, as they result from other dynamics internal to the CSU supplier and on the transport speed and availability, for example. Also notice *e_arrive()* can be thought as the beginning of a subsequent *live state* which in fact represents the UNLOAD activity. This will let us later substitute the event *e_arrive()* by the event *e_startUnload()*.

7.2 The stocking process

We considered the *stocking process* the gathering of two distinct processes: the *input process* and the *output process*. As their names state, the former is responsible for inputting material to the stock, and the later for the output of material from the stock. Figure 10 gives an idea of the events and activities involved in the *output process*:

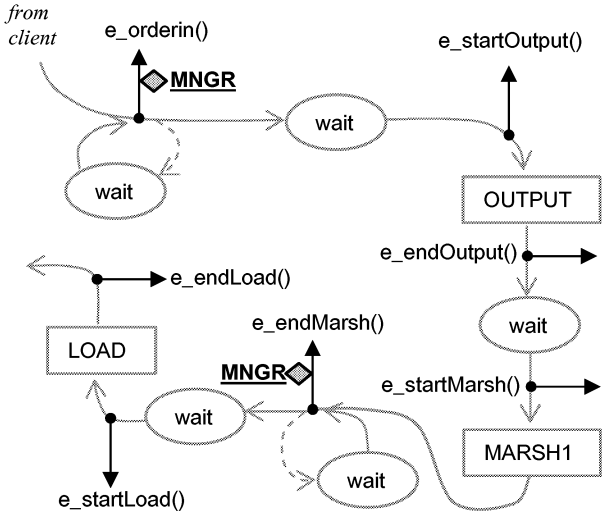


Fig. 10 State diagram of the stocking output process.

Basically, an order-for-output from a client enters the CSU by means of an *e_orderin()* event. This event will then ask permission to an upper-level "*manager*" for the satisfaction of such an order. The "*manager*" will decide if the order will be immediately served, rejected as *stockout*, or made to wait to be satisfied later when new material will arrive from the supplier. As long as the order can be served, the process enters in the phase of outputting the material from the stock. Then the material will be prepared and packed at the MARSH1 activity, and the order becomes ready to be sent to the client. Anyhow, the "*manager*" called by the *e_endMarsh()* event will decide whether or not the order wait for later LOAD and transportation, keeping it in a different waiting place depending on such a decision. The event *e_endLoad()* will trigger the *delivering process*.

The *input process* is basically the inverse of this process. However, in the present approach the *input process* is considered reduced to the activity of UNLOAD, which is due when new material arrives in a vehicle from a supplier. This is the same as considering the material will be available in the stock precisely at the start of the UNLOAD, which great simplifies the modelling. In fact, as not-yet-fulfilled orders will have to wait in the queue before OUTPUT activity, this approach is expected not to lead to significant discrepancies in the results, as it is as if the *input process* would, in the worst case, run in parallel with the *output process* and then, in average the material will already be in stock when it must be taken out. This consideration was mainly used to simplify instantaneous holding costs calculation during the simulation process.

7.3 The delivering process

This process is responsible for carrying the material to the clients. As it was referred earlier in this paper, this can be done by many different ways, using from normal roads to airlines, and thus the efficiency of this process will be great dependent on the kind of transport it assumes. Although the delivery process is conceptually very simple, as it is presented in figure 11, it can turn much more complex in certain cases, for example, when using paths of *transfer* vehicles between two nodes or when the delivery paths are complex. Anyhow, the vehicle entities will be responsible themselves to handle all the material while they move along the network, what will make them practically transparent to the delivery process as it is defined in the present figure.

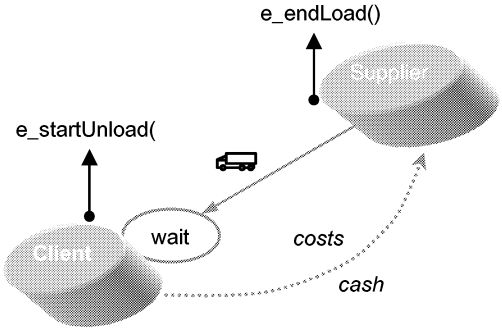


Fig. 11 The basic delivering process.

Thus, one considers that this process begins with the event *e_startDelivery()*, which is equivalent to *e_endLoad()*, and will end in the moment the vehicle will be received by the CSU client at the event *e_startUnload()*. In the case of more than one vehicle on delivering to the same client, probably there will be the need to wait for a free *inputBay*, as figure suggests. With this model only during or after the simulation one will be able to estimate the *lead times* of the system. Unlike certain Supply Chain Simulators, here *lead times* become outputs from the simulation, and not inputs. Nevertheless, the time spent on the return of the vehicle to its home is estimated, considered to be 75% of the time consumed to reach the client. At the end of this process costs and other parameters are computed, and the respective *cash* made to flow to the right suppliers.

7.4 The manufacturing process

The *manufacturing process* is modelled here assuming no product mixing and only one line of production for each product manufactured. This, of course, can be made more complex in future, but at the moment it will be treated in this simplified form, as our main concern is to focus the attention in the overall supply chain behaviour and not particularly in production. In a wide range of real cases, however, this model can be used without affecting significantly the results. The state diagram of such model is presented in the next figure (Fig. 12).

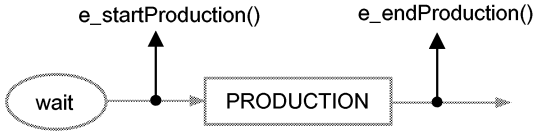


Fig. 12 The basic manufacturing process.

There is a waiting state (queue) for the orders waiting to enter the production, and the events *e_startProduction()* and *e_endProduction()* signalling the start and the end of the PRODUCTION activity. This model also uses the classical concepts of *production rate*, which establishes the maximum frequency the orders can be made to enter the PRODUCTION activity, and the *raw cycle time*, what defines the time spent by each lot in that same activity.

The start of the *manufacturing process* will be triggered or by

the *stock level* of the product to be produced, or using a *cycle* as in the case of continuous production. This will be established by the user. At the proper time, the necessary products and the respective amounts are removed from the stock (using the *output process*) so that the new product can be produced by the *manufacturing process*. At the end, the new product will be added to the stock of the CSU and the related costs computed.

8. About the operational policies in the CSU

The overall activity of the CSU is also dependent on the operating policies adopted, being the most decisive of them the *inventory policy* (how to order the materials), the *production policy* (how to produce), and the *delivery policy* (how to bring the materials to the customers). In this section we briefly present some considerations about the policies implemented in the scope of this modelling approach. To handle such tasks each CSU C++ object includes a *CSUManager()* method to concentrate the code associated with these policies.

The *delivery policy*, for instance, belonging to the FLEET-MANAGER section of the *CSUManager()*, is in this first version very simple: each time the material must be carried to the customer, a vehicle, if any available and free, is loaded with the material and sent to the customer. Otherwise, it will wait. There is not yet *multi-drop* delivery implemented, although different products can integrate the same order.

Concerning the *production policy*, it is given to the user the following three options of managing: ON-DEMAND, if the production is only to start when there is an order from the client which cannot be satisfied directly from the actual stock. BY-LEVEL, if the production is triggered by the level of the stock. BY-CYCLE, when the production is to be continuous and restarted cycle by cycle.

Finally, in the *inventory policies* one can already use a wider range of management options, which go from the simpler (r, Q) and (r, S) models till the notions of *Safety Stock*, KANBAN, Economical Order Quantity (EOQ), and a more general (r(t), Q(t)) model which is expected to try to adjust *r* and *Q* to the demand pattern along the time. In fact, the actual main core of the CSU management is this set of policies, which can be used by the analyst to test different management strategies and other concepts within the model. More about this issue will be presented in a paper to come.

9. RESULTS

The main results of this approach is a software application focused on *Supply Chain Simulation* with which is possible to compare different scenarios in terms of some common Supply Chain Metrics. With such a simulator one can also test operational policies in response to different patterns of demand, in order to have an idea on how fast the chain is able to adapt to demand changes, thus getting an indirect and qualitative measure of its flexibility, as well as of the parameters that can interfere with such flexibility. In fact, this approach also includes a first trial to measure and quantify FLEXIBILITY, or AGILITY, although we only expect to refer to such a subject in a future presentation.

At this stage, we will finish by presenting a simulation of a relatively simple model which is usually used by the *University of Cranfield* (UK) as a didactic supply chain management exercise[4], considering only one product in the chain and all the delivery times set to be one cycle.

Figure 13 shows the network structure used on this exercise, named "*Cranfield Blocks Game*" (Richard Saw 2002). Usually, this game is played with seven groups of students, each group being responsible for the management of a single facility. The customers demand (*Cj*) is random generated, and the supplier (up on the figure) is considered an infinite source of materials. Thus, the groups are only endorsed to manage from the *Depot1* to the *Factory*. No backorders are considered, and each facility has to

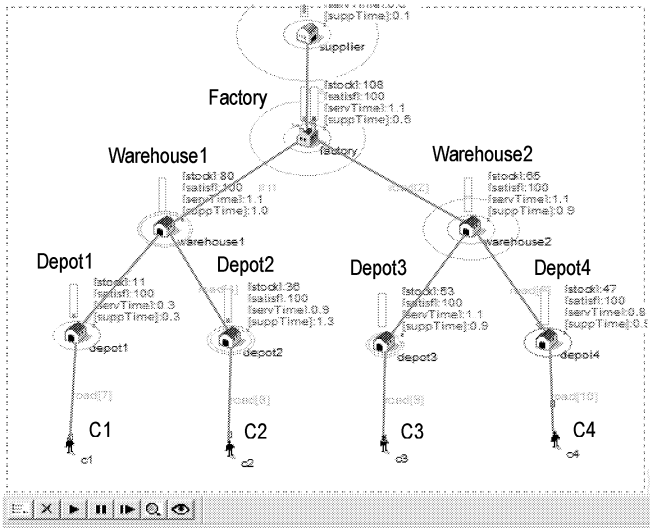


Fig. 13 Cranfield Blocks Game, being simulated.

adjust its reorder quantity to a multiple of a certain *Base Quantity* (BQ) which increases as the facility approaches the *Factory*. All the activity in the chain begin with the customers demand (*Cj*), and the goal of each facility manager is to fulfil demand keeping the *stock* as low as possible without incur in *stockouts*. The average level of costumers demand increases from left to right in the chain, meaning $C1 < C2 < C3 < C4$.

As merely didactic results we present now some output data achieved with the simulation of this case, using in each facility the (r, Q) inventory control model.

The first chart (Fig. 14) shows in different colours the DEPOT3 *local* and *in-motion* stock levels along the time, as well as the demand, the quantity of material arrived from the supplier and also the quantities of the stockouts.

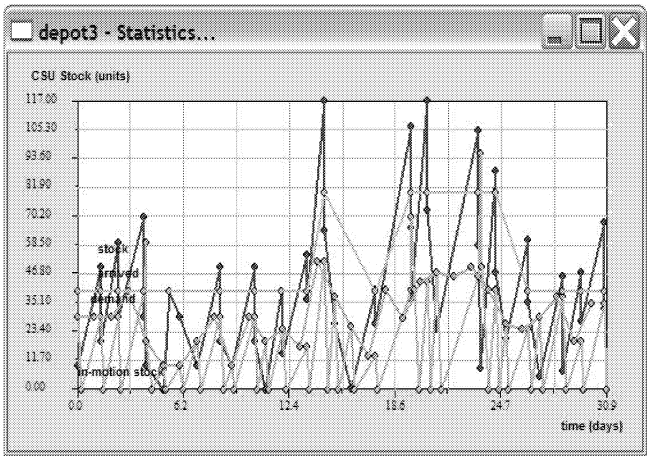


Fig. 14 Depot3 stock levels versus time.

Then, in the next figure (Fig. 15) it is plotted how the WAREHOUSE2 *variable costs* were running with the time. It shoes that was observed a certain stability in the purchasing costs,

in the delivery costs and in the holding costs, comparing with higher amplitude variations observed in the stocking costs due to the input and output of materials from the stock. However, these results have been obtained considering default values in the stocking process, what gives them no special significance. *Accumulated costs* are also computed during the simulation run, letting the user have an idea of the total amounts involved in the CSU activity during the simulated period of time, and can be observed in another sort of chart, not presented here.

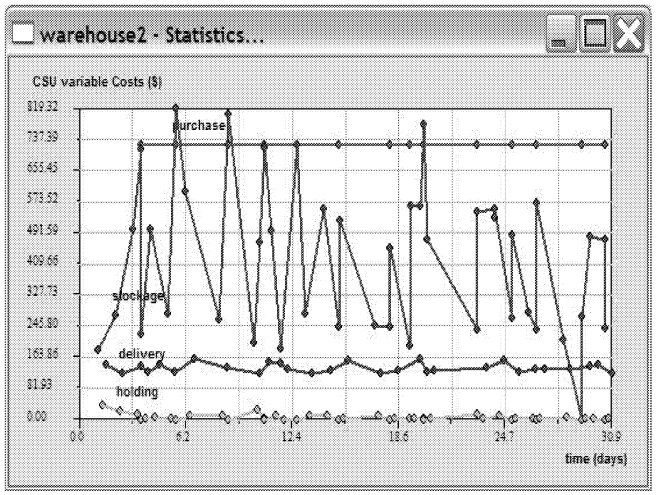


Fig. 15 Warehouse2 variable costs versus time.

The figure 16 relates with WAREHOUSE1 and shows the variations of the supplier's lead time compared with the average time to serve clients. It also shows the delivery time, here considered to be equivalent to the transport time. Based on this information the analyst can better estimate how much the behaviour of such times can affect the performance of the CSU, as well as to project more reliable inventory policies that can better adapt to its suppliers response times. Based on this data the analyst also can understand, for instance, why a certain customer is not being satisfied, whether it is due to its delivery policy or due to its suppliers lead time pattern.

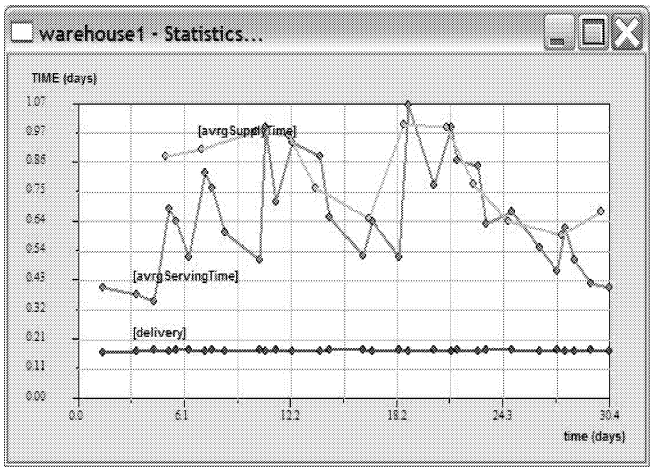


Fig. 16 Some Warehouse1 times versus time.

Finally, figure 17 shows some measures used as Supply Chain Metrics, in this case associated with the WAREHOUSE1 performance computed after 5 simulation runs. On this char is represented the estimated average values and also the variability of measures like TotalCosts, Incomes, Turnover, ServiceLevel, StockoutRatio, etc., thus giving the user the final ability to evaluate different policies implemented in the CSUs, as well as the capability to compare the performances of the different partners in the chain.

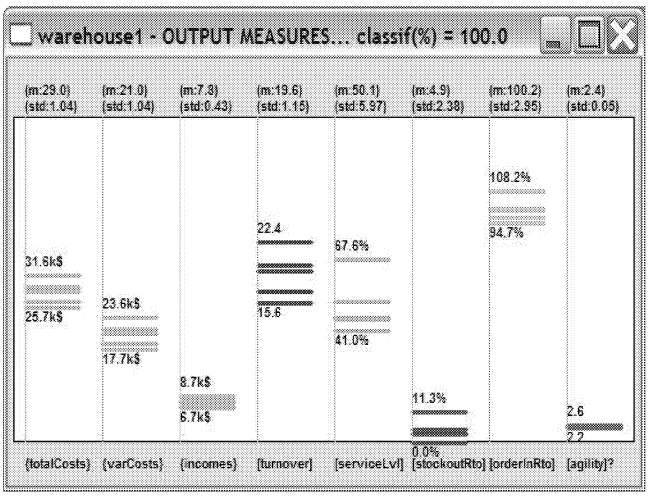


Fig. 17 Warehouse1 output measures after 5 runs.

References:

[1] John E. Sussams, *"Logistics Modelling"*, Pitman Publishing, 1992, p. 55

[2] José Fernando Gonçalves, "*Gestão de Aprovisionamentos*", Edição revista, Publindústria, 1999

[3] António E. S. Carvalho Brito, J. Manuel Feliz Teixeira, *"Simulação por Computador"*, Editora Publindústria, Junho 2001

[4] Richard Saw, "*Cranfield Blocks Game*", private communication, Centre for Logistics and Supply Chain Management (CSCM), University of Cranfield, 2002

TOOLS FOR ENERGY SIMULATION

SIMULATION OF DISTRICT HEATING SYSTEMS FOR EVALUATION OF REAL-TIME CONTROL STRATEGIES

Fredrik Wernstedt, Paul Davidsson and Christian Johansson
Department of Software Engineering and Computer Science
Blekinge Institute of Technology
372 25 Ronneby, Sweden
E-mail: {fredrik.wernstedt, paul davidsson}@bth.se

KEYWORDS
Industrial control, Interactive simulation, System dynamics, Energy.

ABSTRACT

This paper describes a general simulation tool for district heating systems. The purpose is to study the performance of different real-time control strategies for district heating management. A short description of the basic principles of district heating systems as well as some of the managerial considerations is given. Simulation is commonly used within the domain of district heating, both as a strategical decision support tool and as a operational optimisation tool. However, since approaches to distributed real-time control strategies are very rare within the domain of district heating, current simulation tools lack support for such studies. To improve this situation a simulation tool has been developed which integrates existing models of the various entities in a district heating system in a novel way.

1 INTRODUCTION

The optimal operation of District Heating and Cooling (DHC) Systems has for a long time been a important issue for operators of DHC systems. The optimal operation usually refers to the determination of the optimal supply temperature for the nearby future, i.e., from a couple of hours to a few days, depending on the type of production sources and the size and the topology of the supply network. However, we believe that optimisation efforts regarding the production, e.g., (Bøhm et al. 2002, Aringhieri et al. 2003, Arvastson 2001), might loose some of the potential impact because of the stochastic nature of the consumption and the fact that the optimisation models assume a fixed and given demand. The purpose of the ABSINTHE (Agent-based monitoring and control of district heating systems) project [ABSINTHE-WWW] is to deal with this uncertainty by investigating the possibility of using distributed software agents to perform load balancing and local regulation of consumption. The goal could be seen as trying to prolong the validity of a computed optimisation schema despite discrepancies between predicted and actual consumption. This will result in both energy saving and the possibility of delivering a higher quality of service to the customers.

As there are large risks involved in making experiments in a real DHC system, we have developed a district heating simulator in order to evaluate different management strategies. In fact, simulation is a commonly used decision support tool for design of district heating networks as well as for determination of supply temperatures. However, since approaches to distributed real-time load balancing and regulation of individual customers are very rare, current simulation tools lacks the ability to provide environments for controlled experiment of such strategies. This paper describes such a dynamic simulation tool which integrates existing models of the various entities in a district heating system in a novel way.

In the next section, we give a description of the parameters to consider in a district heating simulation model, and in section 3, we describe the novel simulation approach. Finally, we conclude and discuss future work in section 4.

2 SIMULATION OF DISTRICT HEATING SYSTEMS

The purpose of simulation models of district heating systems is to capture the time dependent behaviour of the system. As shown in Figure 1, depending on the focus of the study, many different time scales for simulations of district heating systems can be used (Valdimarsson 1993). At one end we have the transient behaviour at the other we have studies of strategical concern.

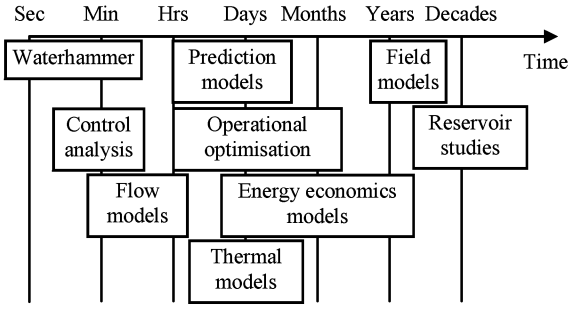


Figure 1: Time Resolution Depending on Focus of Study

In addition to differences in time scale, simulation models of district heating systems can be classified in four categories (Valdimarsson 1993):

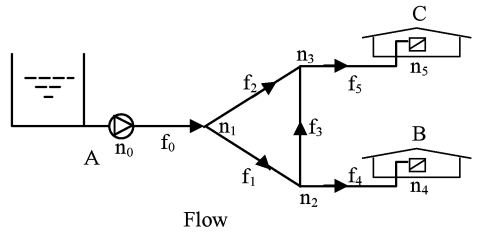
- By *type*, microscopic or macroscopic. *Micro-simulation* generally refers to detailed studies in both time and space of individual entities in the system. *Macro-simulation* refers to studies where the entities of the system are grouped together and modelled as larger entities.
- By *method*, dynamic or steady state. *Dynamic* models calculates current state from previous states, whereas *steady state* models are time-independent or assume steady state conditions.
- By *approach*, physical or black box. *Physical* models uses priori knowledge of the nature of district heating systems, whereas *black box* models are based on relations determined from measured data.
- By *usage*, design or operation. The purpose of *design* usage is to study different configurations of the physical system. *Operational* usage generally considers improvements of an existing system with respect to its economy or performance.

One of the most important components of a district heating system is the network of distribution pipes. When modelling this structure the work of Kirchhoff (Laws of Closed Electric Circuits, 1845) concerning "electrical circuits" is often used. A common way to represent a complete description of the structure is by graph theory. A graph contains information about flows (branches, edges), nodes (connection points) and the relation between them. For electrical networks graph theory is the mathematical basis (Christofides 1975). The first law (the current law) means, that the sum of all mass flows in a node is zero. The second law (the voltage law) says, that the sum of the pressure losses in a closed path (loop) is zero.

Consider a simplified network with 2 loops, and suppose that we want to calculate the steady state. If the current law is fulfilled but the sum of the pressure losses in one of the loops is not zero, a loop correction has to be carried out. This loop correction affects the adjacent loop resulting in yet another loop correction, since this correction also affects the adjacent loops, resulting in yet another correction and so on. One can easily see that the calculation of the steady state of a large system with numerous loops is not easily verified.

A number of techniques based on Kirchhoff's laws has been used for flow calculations of pipe networks. The classical method is the iterative Hardy-Cross method for determination of steady state, or to be used as input for methods such as the Newton-Raphson method. However, this method has drawbacks when it comes to district heating networks. It requires the network to have at least one loop and therefore it can not be used for the most common case, i.e., when there is no loop at all in the system or in the part of the system under study. Valdimarsson has developed a method that solves a matrix equation based on the laws of Kirchoff directly. The approach by Valdimarsson will be described briefly here, more details are found in (Valdimarsson 1993, 1995, 1997).

The general network analysis of Valdimarsson follows the analysis of electrical circuits made by Chua and Lin (Chua et al.1975). An *connectivity* matrix is used to describe the relations between nodes and flows. The matrix has one column for each flow and one row for each node. A starting (1) and a ending (-1) location (node) is indicated for the flow direction in each column. However, it is normally not feasible to describe the network only in terms of pipes and their relations. There are inflows and outflows to the system, or in the part of the system to be studied. These points are considered as boundary conditions of the system (Athans et al. 1974). These locations are places where one know the condition before calculation of the network state, i.e., one knows the flow and pressure. A simple example of a district heating system and the relating connectivity matrix is shown in figure 2.



Node	f_0	f_1	f_2	f_3	f_4	f_5	A	B	C
n_0	1	0	0	0	0	0	1	0	0
n_1	−1	1	1	0	0	0	0	0	0
n_2	0	−1	0	1	1	0	0	0	0
n_3	0	0	−1	−1	0	1	0	0	0
n_4	0	0	0	0	−1	0	0	1	0
n_5	0	0	0	0	0	−1	0	0	1

Figure 2: Connectivity Matrix for a Simple Network. The Shaded Area Indicate Boundary Areas

The connectivity matrix is then rearranged with respect to a spanning tree by splitting it into two submatrices. The choice of spanning tree is based on a certain order of preferences regarding the flow branches. The flow branches are to be sorted according to the following order; head sources, storage tanks, pumps and valves, pipes, and flow sources. One of the submatrices will then contain the flows in the spanning tree and the other will contain the link flows, i.e., the flows outside of the spanning tree. The cutset matrix is then generated from the submatrix containing the spanning tree. A set of branches of a connected graph is a cutset if:

- The removal of the set of branches (but not their endpoints) results in a graph that is not connected.

- After the removal of the set of branches, the restoration of any one branch from the set will result in a connected graph again.

The cutset can be seen as a border that divides the network into two components, that are not connected. Each row in the cutset matrix (see Figure 3) corresponds to a set of flow elements.

A	f_0	f_1	f_2	f_4	f_5	B	C	f_3
1	0	0	0	0	0	−1	−1	0
0	1	0	0	0	0	−1	−1	0
0	0	1	0	0	0	−1	0	−1
0	0	0	1	0	0	0	−1	1
0	0	0	0	1	0	−1	0	0
0	0	0	0	0	1	0	−1	0

Figure 3: Cutset Matrix

According to Kirchhoff's current law the sum of the flow in the flow elements, in each of the cutsets is zero. The cutset matrix is partitioned into submatrices according to the various flow groups, i.e., head sources, storage tanks, pumps and valves, pipes and flow sources. The flow can now be calculated for the network by integrating a state differential equation. Finally, Valdimarsson also shows that a linear theory (Wood et al. 1972) is an effective way of finding the solution of the non-linear steady state by iteration (only necessary for networks containing loops).

3 THE ABSINTHE SIMULATOR

In the following sections the simulation program and its different modules are described. For some algorithms only a brief description is given, more details are provided in the referred articles. We start by describing the requirements, which is followed by an analysis of these requirements. Then the design and implementation of the simulator are presented. Finally, the validation of the simulator is described.

3.1 Requirements

Since the purpose of the simulation software is to evaluate different real-time control strategies in different situations, we identified the following requirements:

- The possibility to simulate arbitrary network configurations.
- The possibility to simulate arbitrary consumer patterns.
- The possibility to simulate arbitrary production strategies.
- The possibility to calculate the flow, pressure and the temperature in all elements of the system as a function of time and the state of the environment.
- The possibility to control parameters during runtime, e.g., possibility to control the consumption and the production during a simulation.
- Recording of the most important output parameters, e.g., mass flow, temperatures, wanted consumption, actual consumption, and pressure of all components.

3.2 Analysis

The focus of the project is to study optimisation of control strategies, hence the time scale of the simulator had to be granular enough to encompass the control analysis time scale (Figure 1). The chosen simulator model is of the microscopic type. The time model was selected in order to fully capture the dynamics of the time dependant behaviour within the individual components.

Components of importance to model are those who may have an active impact on the state of the environment within the district heating system, i.e., those who can be said to control the state. The following control devices need to be simulated:

- Pump stations.
- Heat exchangers.
- Heat producers.
- Storage tanks.
- Valves

The system is based on previously validated models and can thus be said to adhere the physical approach to simulation. Furthermore, the purpose of the simulation is to evaluate different operation strategies, making its usage concentrate on operational considerations rather than design.

3.3 Design and Implementation

In order to achieve a high level of modularity, thus easing future development, the simulator software has been divided into three separate parts:

- Editor, provides the possibility to create a representation of an arbitrary district heating system. The editor also provides an interface for setting all initial values and operational parameters of the simulation.
- Simulator kernel, calculates the dynamic behavior of the system, based on input given from historical states as well as input provided at run time.
- Analyzer, provides the possibility for detailed analyses of output from the simulation.

The simulation program (see Figure 4) comprises approximately 35000 rows of code (approximately 15000 for the simulator kernel) and is written in the Java programming language in order to achieve platform independence. Furthermore, all simulation parameters, component attributes, district heating network configurations as well as the output, are represented in XML, making it easy to add, delete or customize the environment.

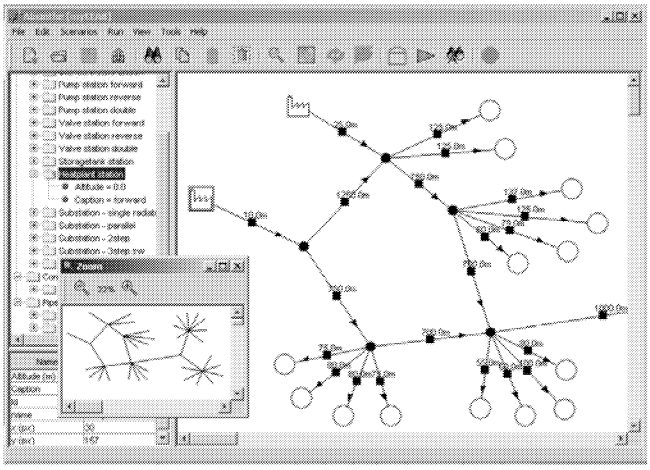


Figure 4. A Screen Shot of the Simulation Tool

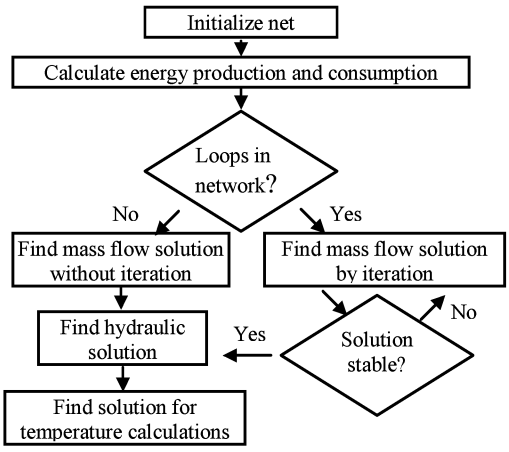


Figure 5: Flowchart of simulation

We will here focus on the simulator kernel. Before the actual simulation starts, the network produced in the editor is divided into subnets. The construction of these subnets is based on the location of components acting as boundary areas. In a very simple network with a single heat exchanger and a single heat producer there will only be two subnets; one for the outgoing flow, and one for the returning flow. In this case, the heat exchanger and the heat plant will act as boundary areas for the subnets. Then, for each of the subnets the following steps are taken:

1. Sort components and build the initial data objects representing the tree, links and nodes of the graph.
2. Calculate the available heat production, the wanted heat consumption and, based on these, the actual heat consumption of this cycle. This stage involves considerations of the possible changes during one single time step to the network state. The initial boundary area values are calculated within the components acting as boundary areas.
3. Calculate the mass flow for each component. If the network contains any loops this has to be done iteratively, i.e., until the threshold limit for the difference between two iterations has been reached.
4. Based on the mass flow we make the hydraulic calculations. The flow and hydraulic values are then stored to the actual data objects representing the individual components.
5. After the mass flow and hydraulic values for all components in all subnets are calculated, the control is returned to the simulator kernel. Based on these new values, the expansion of the temperature wave as well as temperature losses to the environment can be calculated. This is performed by a recursive algorithm traversing the network and advancing the temperature according to the current flow.

At this point the simulation of the current cycle is complete and all that is needed further is to save the values for analysis and for propagation into the next cycle. The flowchart in Figure 5 summarizes a simulation cycle.

In order to simulate the response of management decisions, the ability to prioritise between different heat sources needed to be implemented. Different kinds of heat production models were implemented in the components acting as heat plants. Furthermore, consumption values were needed for simulating the wanted consumption (whether they may get this amount of heat or not is dependent on the actual current state of the simulation). By choosing appropriate production and consumption value sequences we are able to simulate arbitrary states in the network, e.g. heat shortages.

A district heating network is normally composed of a number of additional components, each having their own characteristics and function within the network:

- The *pipe* components usually make up the bulk of the transport system. The mass flow and hydraulic simulation is computed as part of matrix operations on a higher level and not within the pipe component itself. In contrast, the thermodynamic expansion is in part made within the actual pipe component.
- *Valves* are mainly found in conjunction with different types of substations, e.g. heat plants, heat exchangers or pump stations. However, there are no constraints regarding their placement from a simulation perspective. The valve components main purpose is to introduce a certain time delay into the model, thus simulating the time it takes for the real-life components to react. This is important since each step in the simulation is only one second, and a typical valve needs, e.g., ninety seconds to move from fully open to completely shut.
- When considering *pumps* in the simulation, we basically view this component as a negative resistor (Valdimarsson 1993), Thus, in contrast to all other components which add resistance to the fluid, the pump components decrease it. The amount of negative resistance introduced into the system is dependant on the characteristics of the pump, such as maximum mass flow capacity, time to open valves and pressure differences.

- The purpose of a *heat exchanger* is to transfer energy in form of heat from the district heating supply system into the consumer system. This obviously influences both the state of the fluid within the consumer system and within the supply system.
- The *heat producer* is in essence a heat exchanger in reverse; transferring heat from various sources of energy into the district heating transport system.
- It is economically unsound to let the production fluctuate too much. For instance, frequent turning off and on should be avoided. Instead of turning off the heat production when a temporary dip in consumption occurs, the heat can be transferred to a so called *storage tank*. The opposite situation occur when the consumption temporarily peaks. In the simulation model the storage tank acts in a similar way as a heat exchanger, i.e. it either draws energy from the system or inserts it.

The simulation tool provides a great deal of freedom in building hypothetical networks. In doing so one invariably introduce the possibility of the user creating non-realistic networks. This places great demand on the validation process, since there is always a trade off to be made between accuracy and completeness. In this process we have to prioritize the accuracy since we can make greater use of a system that is able to accurately and precisely handle all more or less realistic networks than a system that poorly handles *all* networks. For example, a basic issue was how to correctly represent flow sources and head sources in relation to heat producers and consumers since these change dynamically as the simulation traverses the various subnets within each cycle.

The simulation cycle is one second. However, this does not mean that interaction from the control strategies have to operate at the same interval.

3.4 Simulation Validation and Verification

Valid simulation model components were available and could be combined and calibrated to simulate the performance of a district heating system. The model developed by Valdimarsson for calculation of the steady state has been found valid in comparison with the commercial software package, PIPE-FLO (Valdimarsson, 1993), and for the district heating system of the city of Almere, Holland (Valdimarsson, 1997).

During every phase of implementation we have taken great care in validating the output. All equations has been implemented and solved in Matlab and continuously been used to validate the output from different simulation configurations.

We use the same initial state for different simulations, i.e., all parameters are set according to their initial configuration. The warm-up period varies and depends on network topology and simulation setup.

4 CONCLUSIONS AND FUTURE WORK

We have developed a simulation tool which integrates existing models of the various entities in a district heating system in a novel way. We have used the model developed by Valdimarsson, which is valid for steady state but needed much consideration concerning the environment, to be able to be incorporated into a dynamic simulation of district heating systems. For example, a basic issue was how to correctly represent flow sources and head sources in relation to heat producers and consumers since these change dynamically as the simulation traverses the various subnets within each cycle. Different ways to optimize these processes are under consideration, since they constitute a crucial part of simulating the dynamic environment.

Acknowledgement. We would like to thank the software engineering students in the AgentSimHeating team who implemented a major part of the software during their B.Sc. exam project at Blekinge Institute of Technology. This work has been financially supported by VINNOVA (The Swedish Agency for Innovation Systems).

REFERENCES

[ABSINTHE-WWW] ABSINTHE project web site: www.ipd.bth.se/absinthe.

Aringhieri, R. and F. Malucelli. 2003. "Optimal operations management and network planning of a district heating system with a combined heat and power plant." *Annals of Operations Research*, Vol. 120, 173-199.

Arvastson, L. 2001. *Stochastic modeling and operational optimisation in district heating systems*. Lund Institute of Technology. Doctoral thesis, ISBN 9162848550.

Athans, M.;. M.L. Dertouzous; R.N. Spann; and S.J. Mason. 1974. *Systems, Networks, and Computations*. McGraw-Hill, ISBN 0070024308.

Bøhm, B.; M. Lucht; Y. Park; K. Sipilä; S. Ha; K. Won-tae; K. Bong-kyun; T. Koljonen; H. Larsen; M. Wigbels; and M. Wistbacka. 2002. "Simple models for operational optimisation." Report 2002:S1, NOVEM, ISBN 9057480212.

Christofides, N. 1975. *Graph theory: an algorithmic approach.* Academic Press, ISBN 0121743500.

Chua, L.O. and P.M. Lin. 1975. *Computer-aided analysis of electric circuits*. Prentice-Hall, ISBN 0131654152.

Valdimarsson, P. 1993. *Modelling of geothermal district heating systems*. University of Iceland. Doctoral thesis, ISBN 9979540559.

Valdimarsson, P. 1995. "Graph-theoretical calculation model for simulation of water and energy flow in district heating systems." *In Proceedings of the 5th International Symposium on Automation of District Heating Systems*. (Helsinki, Finland).

Valdimarsson, P. 1997. "Use of a graph-theoretical calculation model for water flow in D/H systems - experience from NUON, Holland." *In Proceedings of the 6th International Symposium on District Heating and Cooling Simulation*, ISBN 9979542039.

Wood, D.J. and C.O.A. Charles. 1972. "Hydraulic network analysis using linear theory." *Journal of the Hydraulics division*, Vol. 98, No. 7, 1157-1170.

SIMREN, A SIMULATION TOOL FOR RENEWABLE ENERGY SYSTEMS

Stefan Herbergs
Harry Lehmann
Stefan Peter
ISUSI, Institute for Sustainable Solutions and Innovations
Römerweg 2, 52070 Aachen, Germany,
Phone: +49 241 4095 68-0, Fax: +49 241 4095 68-68
e-mail: hl@isusi.de

ABSTRACT

The computer-modelled simulation of renewable electricity networks is a method for the research and optimization of electricity / energy systems with a high share of renewable energy suppliers. The SimREN Software offers the possibility to design "close to reality" models of energy supply and demand systems following a "bottom-up" approach. Independent and detailed models for energy demand, energy management, adapted distribution systems and energy supply can be used to study different energy systems relying on renewable sources. A country or island can be divided into 15 regions, each region subdivided in up to 15 sub-regions, each consisting of many different suppliers and consumers. The simulation uses real measured weather data for a complete year with a typical time resolution of 15 minutes for one simulation step. Both supply and demand can be simulated with their dependence from the actual time and weather. With the SimREN System it was possible to study the dynamics of energy systems with a high share of renewable suppliers and to show that a "typical" OECD country could be supplied completely from renewable sources [Spangardt 1999]. The simulation helped to develope and optimise the system in order to make supply reliable and keep the technological expense at a minimum.

INTRODUCTION

Energy systems must guarantee that energy production always is sufficient to meet demand. Electricity must be produced by the time it is needed. Therefore, in energy supply systems with a high share of renewable energy suppliers, the electrical subsystem is the most (time) critical component. To guarantee electricity supply's reliability it is necessary to prove that production rates always meet demand. A dynamic simulation of the electrical supply system can be used for proving and optimizing such supply systems.

THE STRUCTURE OF SIMREN

SimREN is a dynamic simulation tool, which calculates the energy supply and demand with a certain temporal resolution. It has a bottom-up structure. This means it consists of different elementary blocks that are combined to bigger blocks, which - in total - form the model of the regions or islands whole energy system. An elementary block – for example - could be a wind turbine. Several of them can be combined to form a wind park. These wind parks, together with other energy suppliers and energy consumers, can build a logical region of the whole system. The different energy components yet included in the system are shown in the graph below. The graph also shows the assumed energy flow for a renewable energy system.

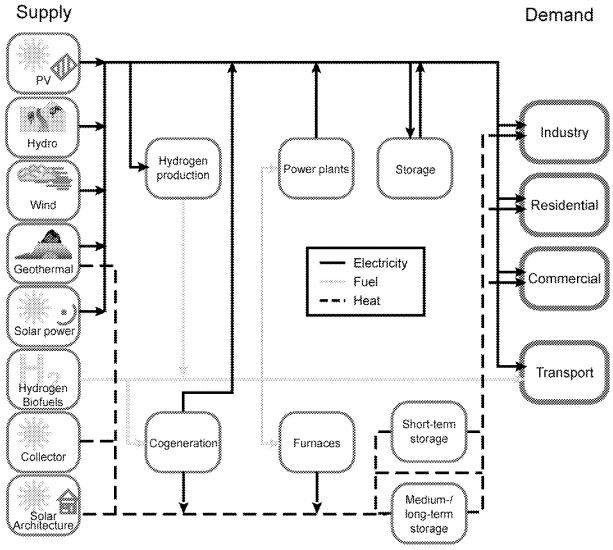


Figure 1: Energy flow in a renewable energy system in SimREN between the included components

The country or island simulated with SimREN can be divided into 15 regions, which can supply each other with energy. If one region cannot supply itself with energy and another region already produces more than needed or is capable of producing more, they can exchange energy. The energy manager, which can be set up for different strategies in energy supply, does the task of exchanging energy.

Each region can be subdivided into ten to fifteen sub-regions, each consisting of many different energy suppliers and consumers. The energy suppliers are categorized as fluctuating or adjustable energy suppliers (non-fluctuating).

The simulation runs for a complete year with a typical time resolution of fifteen minutes for one simulation step, but shorter intervals are possible as well.

The structure is illustrated in the graph below.

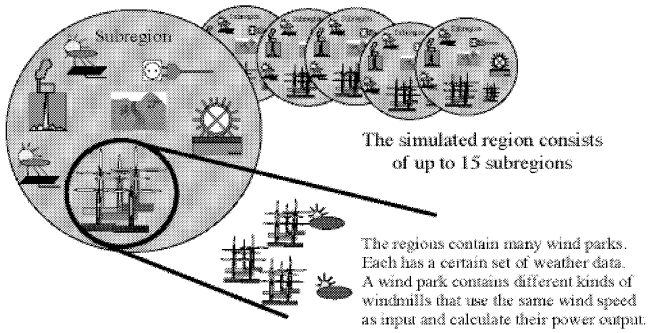


Figure 2: The structure of the SimREN simulation

INPUT DATA

SimRen uses a database of real weather data and detailed information about the installed capacities of energy producers to calculate the energy output of certain renewable technologies. Typical demand profiles of days for the different seasons - that is the variation of energy consumption in the curse of a day - are also needed to calculate the energy demand throughout the year. A persistent algorithm in the simulation, which calculates the energy demand and supply at every time step, uses this information.

SIMULATION

The simulation consists of four parts:

- Energy Demand
- Energy Supply by fluctuating sources
- Energy Supply by non-fluctuating sources
- Energy Manager / Energy Exchange

First of all the energy demand is calculated.

Secondly the electricity production of fluctuating suppliers in every region is determined and subtracted from the energy demand. The remaining demand is what has to be covered by adjustable suppliers and storages, which are last in the simulation sequence.

The energy manager is in control of the adjustable energy suppliers and keeps track of energy production and consumption in order to adjust supply to demand.

DEMAND MODEL

The simulation of energy demand is essential to guarantee a good simulation of an energy supply system. It is very difficult to get measured high quality data of a country's energy demand for every hour of a year. Therefore a demand model must be included, which calculates the energy demand for every hour of the year from typical demand profiles.

SimREN can generate the demand for the whole year from typical daily and monthly demand profiles. An example of such hourly demand profile is shown below [ERJ 2003].

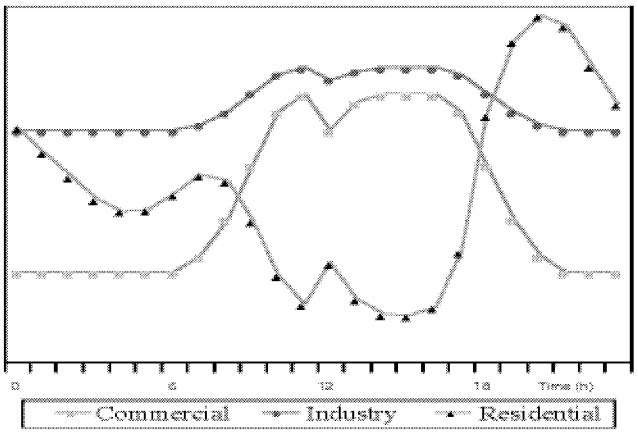


Figure 3: Typical energy demand profiles for a summer working day

A complete set of daily demand profiles (typical working days and holidays of every season of the year) should exist in order to get a properly synthesised and hourly resolved demand profile for the whole year, thus representing the energy demand for every single hour of the year. This kind of data can often be obtained from the energy supply companies. The [UCTE] webpage for example has daily demand profiles for every participating country. Merging the curves results in the given energy consumption for the whole year, containing the monthly and daily profiles on an hourly base. It is necessary to obtain data for the different sectors of energy consumption such as industry, households, etc.

All the above mentioned profiles were available for Germany and for a "typical" OECD-country.

Once having completed the demand model SimRen allows for testing different options in energy demand, such as demand management or the introduction of energy efficient technologies.

SUPPLY MODEL

SimREN's energy supply model consists of fluctuating (e.g. windenergy and photovoltaics) and non-fluctuating sources (e.g. cogeneration plants, hydropower plants, etc.).

The different energy producers can be freely distributed to the regions of the model to keep the installed capacities well adapted to geographical and climate conditions of a region for every technology included.

SimREN exchanges energy between the included regions, if there is an surplus or a deficit in a certain region.

This is illustrated in the figure below.

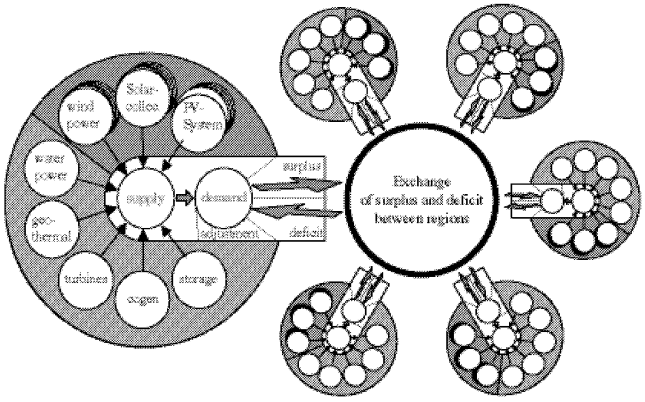


Figure 4: Example of regional energy sources

Fluctuating Sources

Wind energy and photovoltaic are fluctuating electricity suppliers because they depend on the wind and solar radiation respectively [Quaschning 1999]. In addition to these two suppliers, cogeneration plants in the residential and commercial sector can be seen as fluctuating energy suppliers (depending on outside temperature) as the heat needed in these sectors determines the production of electricity. This is different if they are handled as virtual power plants.

Non Fluctuating Sources

Adjustable energy suppliers used in SimREN include hydropower plants, geothermal power plants, fast reacting power plants and cogeneration power plants for high and low temperature heat. The power output of pumped storage plants is also adjustable but restricted by storage's water level. Because the maximum power output of the hydropower plants fluctuates with water level of the rivers, the controllable maximum power output is restricted to that.

Energy Manager

The energy manager of the simulation takes control of the adjustable energy suppliers. This energy manager controls the cogeneration plants, the hydropower plants and the geothermal power plants.

Due to the modular design, the energy manager can follow different strategies in energy supply.

A short example:

Besides producing electricity, cogeneration plants can supply heat at different temperatures (low- or high-temperature heat). If the cogeneration plants are set up to cover heat demand, the simultaneous produced electricity can be used for public supply, increased independece from the public grid and/or the production of hydrogen. If they are adjusted to meet the public electricity demand, the simultaneous produced heat can be used for industrial processes, district heating, etc.

Having control of the adjustable energy suppliers – including storages and backup capacities -, the energy manager can ensure the most rational use of the different technologies according to regional boundary conditions.

The energy manager also has the capability of distributing energy between the different regions. If, for example, one region has got a deficit in energy supply, the energy manager will ask the other regions to fill up the gap. Doing so, the energy manager keeps track of distances between the regions and grid losses caused by transportation to choose the most rational option.

RESULTS

The results of simulating a "typical" model OECD country showed the possibility of supplying the energy needs solely from renewable sources [ERJ 2003]. While the energy demand was reduced by the introduction of energy efficient technologies in all sectors of energy consumption, the supply system's technological mix was optimized regarding least amount of installed capacities and greatest reliability in supply. The high spatial resolution gave the opportunity to benefit from specific regional offerings of renewable sources and thus to minimize fluctuations in energy production. The need for backup capacities was minimized as well.

Several simulation runs were used to optimise the system regarding supply reliability throughout the year. After that the energy supply showed to be as reliable as conventional energy supply systems.

Results of a simulation with SimREN are exemplary shown in the graph below

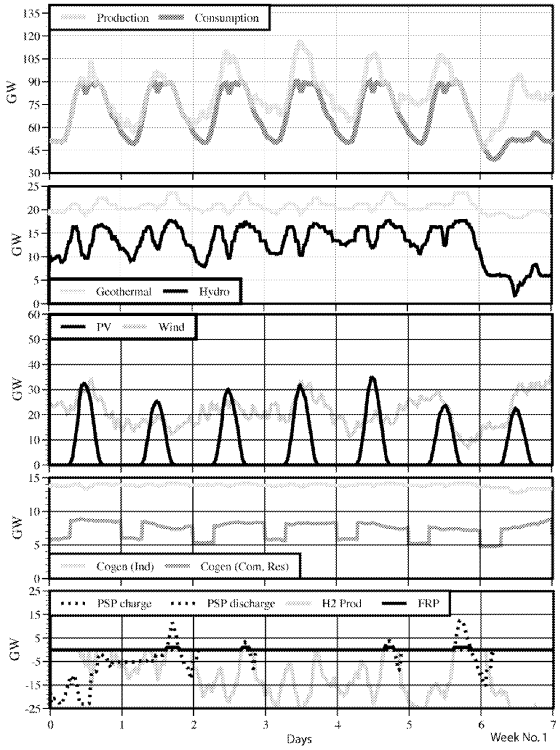


Figure 5: Energy supply and demand of a "typical" model OECD-country for one week

To illustrate the results of SimRen, the graphs above show the energy production and consumption balance for the first week in January of a simulated 100% renewable energy supply for an OECD country. The first graph (above) shows the energy consumption compared with the production, while the other graphs show how the full supply was accomplished. The second graph shows the adjustable energy suppliers and how their output is regulated to compensate for the fluctuating energy suppliers PV and wind (3rd graph) at times of low production. The 4th graph indicates the heat demand in the residential and commercial sectors, where the cogeneration output can be seen to correlate with the outside temperature. Even powering down the systems during night times is represented by this graph. The output of industrial cogeneration is relatively stable throughout the week. During day 1 and day 5, the discharge of Pumped storage plants can clearly be seen to coincide with a relatively low production of PV and wind (3rd graph). The last graph also contains energy that was needed to fill pumped storage plants and to produce hydrogen.

Graphs of this type, showing the results for every day of the year, also exist for each region of the simulation. The detailed results give the opportunity to analyse regional conditions in depth. This information is also helpful in identifying "hot spots" for future action and thus in developing strategies for extending the share of renewable energy sources

FUTURE WORK

The future development of SimREN focuses on an extended functionality of the database and improved data handling. One of the next steps will be using SimREN to develop renewable energy supply scenarios and strategies with a focus on fuly renewable energy supply.

BIBLIOGRAPHY

[ERJ 2003] Energy Rich Japan
ERJ Research Team

[Quaschning 1999] Regenerative Energiesysteme
Volker Quaschning

[Spangardt 1999] Entwicklung und Anwendung eines modular aufgebauten Simulationsprogramms zur zeitlich und regional aufgelösten Untersuchung regenerativer Stromversorgungssyteme
Gordon Spangardt

[UCTE] www.ucte.org

TOOLS FOR ROBOTICS SIMULATION

VirtualRobot: An open general-purpose simulation tool for Robotics

Martin Mellado, Carlos Correcher*, Juan Vicente Catret* and David Puig*
Departamento de Ingeniería de Sistemas y Automática (DISA)
Universidad Politécnica de Valencia (UPV)
Camino de Vera, s/n, 46022 – Valencia, Spain
Phone: +34-963879570, Fax: +34-963879579,
vrs@isa.upv.es http://www.isa.upv.es/~vrs

* Currently working in *Institute for the Protection and Security of the Citizen* (IPSC), Joint Research Centre – Ispra.

KEYWORDS

Robotics simulation, simulation tools, education by simulation, research by simulation

ABSTRACT

This paper introduces the basis of a graphical software application, designed for low cost hardware, as a general, flexible and open platform to work on robotics. *VirtualRobot* has proven to be a useful tool in many fields of robotics, such as manipulator programming, mobile robots control (wheeled, sub-aquatic and walking robots), distance computation, sensor simulation, collision detection, motion planning and so on. *VirtualRobot* is used as the common interface for all the applications. Some practical cases and application examples are illustrated on the paper to present the possibilities of this simulation software.

INTRODUCTION

Nowadays, graphic simulators for robotic systems are indispensable in most of the robot design, learning and exploitation steps. Since this kind of applications were implanted during the eighties (Stauffer 1984; Howie 1984; Parker 1985), technological advances and improvements in computing engineering allow them to be applied on any field in robotics: industrial robotics (Freund et al. 1999), mobile robotics (Yang et al. 1997), sub-aquatic robotics (Chen et al. 1997) or aero-spatial robotics (Kazuya 1995). In addition, they have become significantly more powerful and flexible.

Some simulators go further than usual 3D graphical representation, applying virtual reality techniques (Matsumoto et al. 1999; Pérez and Schilling 2001). Sensor data inclusion allows more advanced simulators (Kimoto and Yuta 1995; Hale and Gosine 1999) to represent, in a dynamic way, robot environment as detected by sensors. With these methods, simulation software is not only dedicated to simulate robot behaviour, but also is useful for design, analysis and validation of techniques as collision detection, motion planning, sensor modelling, evaluation of control architectures and so on.

In most of the cases, they are very complex simulation systems (usually with expensive hardware requirements) designed to study and analyze a punctual characteristic within a specific branch of robotics. Therefore, there is still an open subject for future simulation software with general features and open software architecture, less specific and more flexible, to be adapted for most topics in robotics education, research and industrial implantation.

In this paper, the basis of a graphical software application as platform to work on robotics is explained. *VirtualRobot*, created originally for remote robot monitoring, programming and simulation of a specific robot control system, has become a useful general tool in many fields of robotics, such as manipulator programming, mobile robots control (both, wheeled and walking robots), distance computation, sensor simulation, collision detection, motion planning and so on. *VirtualRobot* is designed for low cost platforms and used as the common interface for all the applications. On the next section, *VirtualRobot* origin and main features are exposed. Its open software architecture is then explained with special emphasis on its implementation and how client's applications can be integrated on. The last section contemplates practical cases with different application examples.

VirtualRobot

VirtualRobot is a freeware software suite[1], in the sense that includes several programs, developed in C++ by DISA-UPV Robotics Group, for robotics application, research and education, with a graphical representation based on OpenGL under Microsoft Windows (2k/NT/9x) operating systems. Its beginning, on 1998, was motivated by the need of a remote graphical application for programming, monitoring and simulating multi-robot cells on the control system GENERIS (Generalised Software Control System for Industrial Robots) (Ruiz 1999) developed by European Commission Joint Research Centre. Its open software architecture design, explained on the following section, allows, i.e., kinematics modules to be ported to GENERIS control system.

[1] *VirtualRobot* can be downloaded in http://www.isa.upv.es/~vrs

VirtualRobot is composed on:

- A basic geometric modeller *VirtualRobot Modeller* (*VRM*), to create and edit geometric models. Specific ASCII structured file formats are defined for elements to be used in *VirtualRobot*.
- A geometric data translator, *VirtualRobot Translator* (*VRT*) to convert files from AutoCAD® DXF and VRML file formats in addition to a special plug-in module to export data from 3DStudio Max®.
- And the main platform for simulation, *VirtualRobot Simulator* (*VRS*), including some components (a set of adaptable Dynamic Link Libraries) and external applications (*VRS Tools*, *VRM Tools* and *VRS Demos*).

VRS can be applied for simulating any type of industrial robots, individually or grouped in multi-robot workcells for their off-line automatically part-program generation and testing, as well as for on-line programming and monitoring. *VRS* can be connected to a numerical control (NC) as GENERIS using TCP/IP to monitor the process and robots status.

Robots can be included in a user-defined environment composed on static elements such as objects (pure static elements) and parts (can be manipulated with robots). *VRS* can simulate not only manipulator-arms, but also multi-axis machines as conveyors, turntables, machine-tools, sensor systems and mobile robots. Robots can be attached one to one, or many to one, in order to form more complex and redundant devices, such as walking robots.

VRS easy and friendly user interface (Figure 1) notably reduces learning phase, even for new users on this kind of applications, in such way that it becomes adequate for educational, research and industrial purposes (Mellado et al. 2001a). *VRS* has been also successfully used for robot off-line programming on rapid prototyping in industrial applications, thanks to its flexible architecture that allows data integration from and to other applications. This fact allows client's applications to share a common graphical interface making easier to manage different programs. *VRS* can be enriched with user-made software in order to implement specific applications.

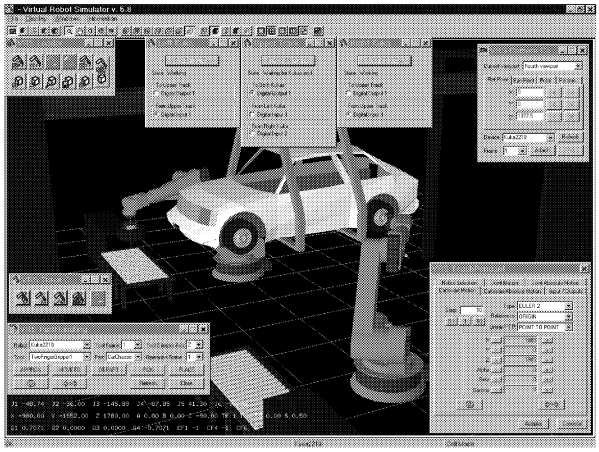


Figure 1. *VRS* Graphic Interface

VRS SOFTWARE ARCHITECTURE

VRS software architecture (Figure 2) is based on the following four parts explained in next subsections:

- *VRS Kernel* including user interface,
- a set of external components *under VRS* that the user can replace with his/her own components according to simple coding and compiling constraints,
- an external access library (*VReal)* for external application development
- and the external applications that run *over VRS*.

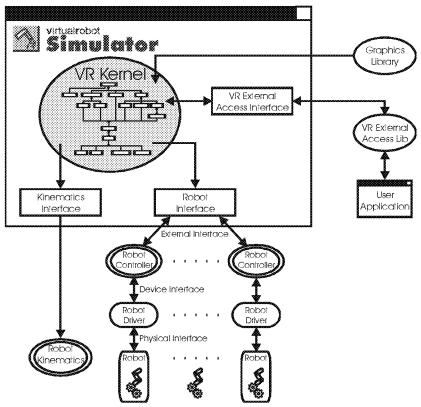


Figure 2. *VRS* Software Architecture

VRS Kernel

VRS Kernel has been designed through a hierarchical structure of classes (Figure 3). This hierarchy has been implemented following the inclusion principle, where an entity is modelled as a set of smaller parts. In this way, an environment geometric model, a robot list and a list of range sensors mainly compose simulator class. The environment represents a set of objects and parts, while a robot is defined by an ordered set of links and optionally an adapter and one or more tools. Each tool is defined by a list of tool status. Finally, an element (object, part, link, adapter or tool status) is defined by figures, that is, sets of geometric primitives.

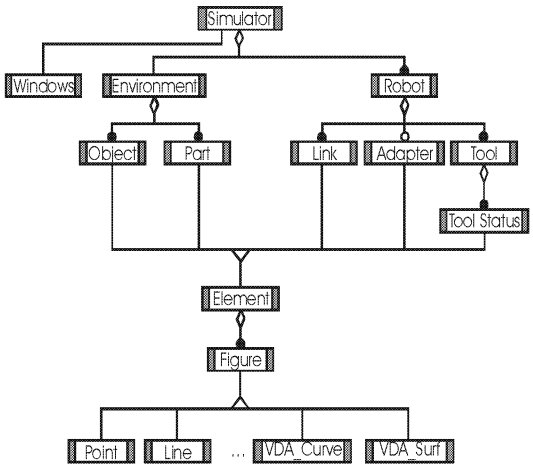


Figure 3. *VRS Kernel* Classes

VRS Kernel implements the graphical display control and handles user action events. When the user action generates an event, a CSimulator class method (or methods) is called. This display control is only the graphical interface, that is, for example, no robot can be loaded or managed through this interface. As commented in next sections, real user interface is supplied with external applications.

VRS Kernel is composed of objects data and processing threads. The most relevant processing threads are the following:

- Main application thread. This thread controls the user interface. According to the user interaction, modifies class data or shoots draw event. It also controls the visual display including viewports, render mode, camera control, …
- Drawing thread. There is only one thread of this type. This thread is the unique thread that can draw on the screen. The thread is waiting for a draw event that can be generated by the simulator class or by the user interface class. The thread obtains data from the classes, configures the screen and draws the visual components of the scene.
- Local communication control thread. There is only one thread of this type. This thread is waiting for a local communication request generated by *VReal* in local mode and starts a local communication thread. Local mode is based on shared memory.
- Local communication thread. There is a thread of this type for any external application connected to *VRS Kernel* via *VReal* in local mode. These threads are waiting for a local communication order request event. This thread makes the following actions:
  - receives order command and input parameters,
  - generates a simulator method call with this parameters,
  - obtains and sends the output parameters to the external application,
  - returns the order execution result to the external application,
  - and, if required, generate a draw event.
- Remote communication control thread. There is only one thread of this type (triggered by the user). This thread is waiting for a remote communication request generated by *VReal* in remote mode and starts a remote communication thread. Remote mode is based on TCP/IP communication (using sockets interface).
- Remote communication thread. There is a thread of this type for any external application connected to *VRS Kernel* via *VReal* in remote mode. These threads are waiting for a remote communication order request event. This thread makes the same actions than the local communication threads.
- Range sensor scanning thread. There is a thread of this type for any range sensor started on the simulator. Any of these threads is computing distances from the sensor to every element in the scene.
- Collision detection thread. This thread (when active triggered by the user) is shot when any component in the scene has been moved or modified. It checks all possible collisions between mobile elements (robots) and mobile or static elements (robots, objects or parts). When a collision is detected, background color is modified, robot motions are stopped and a draw event is triggered.

VRS external components

The external components are Dynamic Link Libraries (*dlls*) loaded on memory during execution time. The user has a set of code templates available for developing new components that can be used instead of the original ones, in such a way that *VRS* can be adapted to the user requirements. The most relevant external components are:

- Vector and transformation components. These components implement vector and transformation classes. Standard components are included in *VRS*, which probably will never be modified even if this is possible.
- Kinematics component. This component solves forward and inverse kinematics for robots. Most of the industrial robots and even some redundant robots are already solved with the original kinematics component. New mechanical structures can be easily studied just developing its specific kinematics module. For any robot loaded on *VRS*, a different kinematics component can be selected.
- Robot control component. A robot control component allows to send commands to real robots connected to *VRS* in such a way that any order send to the simulator from an external application will be executed firstly on *VRS* and if no error is produced, executed then in the real robot. Two robot control components have been developed so far for the robots available in the DISA robotics laboratory using serial RS232 communication. New robots can be easily integrated on *VRS* by just developing the proper controllers to link the applications to their drivers. For any robot loaded on *VRS*, a different robot control component can be selected.
- Distance-computation component and sensor-model component. With the help of these components, range sensors can be modelled, implemented and included in *VRS*. The first component realizes the proper distance computation solving a ray tracing problem: compute interfering situation, distance and interfering angle related to Z-axis of a sensor location to every element figure in *VRS*. The sensor-model component allows modifying distance, interfering angle and returned reflectance according to a specific sensor model.

These components give great opportunities to adapt easily *VRS* to the user requirements.

VirtualRobot external access library (*VReal*)

However, the most powerful tool to adapt *VRS* to user requirements is the *VirtualRobot External Access Library* (*VReal*). *VReal* is implemented as a Dynamic Link Library in order to make possible the interaction among client's application and *VRS*. In fact, *VRS* is a basic platform with graphical and special robot capabilities that offers functionalities to the user programmer through *VReal*.

VReal provides an interface formed by a set of more than 250 C/C++ functions (in addition to constant and type definitions), whose objective is to allow client's applications managing elements defined on *VRS* (robots, range sensors and environment elements) and even creating new elements on *VRS*, such as objects generated from sensor data. The functions are organized into independent sets:

- Functions to initialize and close *VReal*
- Functions to manage robot and environment files
- Functions to control robots in *VRS*
- Functions to handle auxiliary figures that the client's application can manage on *VRS*
- Functions for *VRS* user interface functionalities, mainly to configure view parameters (zoom, point of view, reference view point, …)
- Functions for distance computation and collision detection
- Functions for range sensor control

VReal can be compiled using a static or dynamic link in client's applications. The communication between a client's application and *VRS Kernel* is made in the following way:

1. *VRS Kernel* has a special communication thread waiting for a connection request
2. A client's application must initialize *VReal* with the function provided for this purpose asking for a connection
3. *VRS Kernel* processes the request and creates a communication thread for the client's application. From this moment, the client's application has access to all the functionalities available on *VReal* through this communication thread
4. *VReal* must be closed to finish communication
5. New connection sessions can be started according to steps 2-4.

There are two working modes in *VReal*, shared memory mode and TCP/IP based mode. Due to the internal implementation structure of *VReal*, only the basic communication module must be adapted to be used in different operating system platforms.

External applications

As commented above, *VRS* is the basic platform for external application simulation and display. Therefore, *VRS* user interface give no possibility to create or manage scene elements; it is only designed for graphical interface. In fact, the user interface is one of the functions to be done by external applications. The external applications form the real user interface of any specific application. On the other hand, it is important to notice that any element created in *VRS* by an external application is stored on *VRS* and will be accessible by any other external application. Using this feature, external applications have been developed to serve as user interface for *VRS*. Once again, the user can adapt *VRS* to special requirements, constructing the required interface as new external applications. As an example, default external applications included in *VirtualRobot* suite are:

- *VRS Tools*: A set of external applications that extends the user interface to robotics purposes. The default *VRS Tools* with their main features included in *VirtualRobot* suite are listed in the table below.
  - *VRS Loader*: for file management: robot and environment load, place and close
  - *VRS TeachPendant*: for robot control: robot selection, joint and Cartesian motion, tool actions, robot speed control, I/O digital and analogical signals, …
  - *VRS TraceControl*: for robot trace control
  - *VRS IOConnection*: for digital and analogical signal connection between robots
  - *VRS VideoRecorder*: for video recording
  - *VRS PartHandling*: to pick and place parts with robots
  - *VRS SpeedControl*: for global motion speed control
  - *VRS CameraControl*: for camera control and attachment to robot frames, cameras, ...
  - *VRS TrajectoryControl*: for robot trajectory execution
  - *VRS CollisionDetector*: to start or cancel collision detection
- *VRM Tools*: A set of external applications that forms the user interface of *VRS* for modeling purpose.
- *VRS Demos*: A set of external applications that shows the basic possibilities of *VRS* in robotics.

Note that external applications are just executable programs that connect to *VRS*. They can be executed from the operating system or from *VRS* menus provided for this purpose.

APPLICATION POSSIBILITIES

As commented on introduction, *VirtualRobot* suite was initially designed for manipulators simulation. But in this section, after commenting the use for this kind of platforms, new trends will be introduced when applied on mobile robots and sensors simulation.

Application for manipulators

From the basic platform, and using the important adaptability feature that *VRS* has, many works have been developed for manipulators. As examples, the following applications and their utility can be listed:

- Specific application development for industrial process simulation: part handling, spot and arc welding, assembly, palletizing, painting, drilling, ... Some examples are included in *VirtualRobot* suite (Figure 4), both, for single or multiple robot processes. The development of this kind of application is useful for teaching, program depuration (motion locations, I/O signal protocols, program sequence control and so on), process analysis before implantation, etc. An application of *VirtualRobot*, in the industrial field, has been the implementation of a CAD/CAM system for rapid prototyping in the automobile sector. CAD data in VDA format is, in some steps, translated to an ABB Rapid program for ABB IRB 2400 execution, allowing different design cycles.

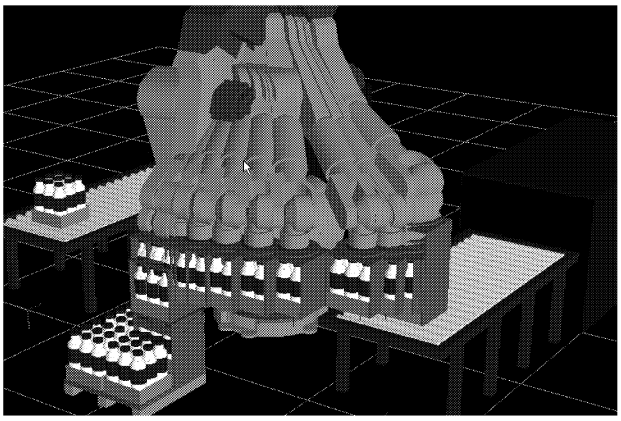


Figure 4. Industrial Process Simulation (Palletizing).

- Robot programming languages interpreters and simulators. Two interpreters have been implemented, a CRS Grapl-II (Mellado et al. 2002) interpreter and an ABB Rapid interpreter. These interpreters can be used again for teaching purposes as well as for implantation analysis and depuration.
- 3D Input device analysis. In a research project about virtual reality techniques applied to robotics, *VirtualRobot* has been used to evaluate the use of different 3D input devices (Mellado et al. 2001b) and a sensor based haptic device (Zotovic et al. 2001).

In addition to this applications base on the implementation of external programs running *over VRS*, another important application was based on the use of the components *under VRS*:

- The need for developing and evaluating robot kinematics software module for the GENERIS NC robot control system has led to the development of a kinematics component on *VRS*. In (Puig et al. 2001), the use of *VirtualRobot* as a general development and testing system for robot kinematics methods was presented. Its application to the complex case of a configurable redundant (9 degrees of freedom) robot manipulator (Figure 5) designed as a prototype system for the maintenance of thermonuclear fusion reactors, laser welding of large components and remote inspection of nuclear storage areas has served to demonstrate the features of the system.

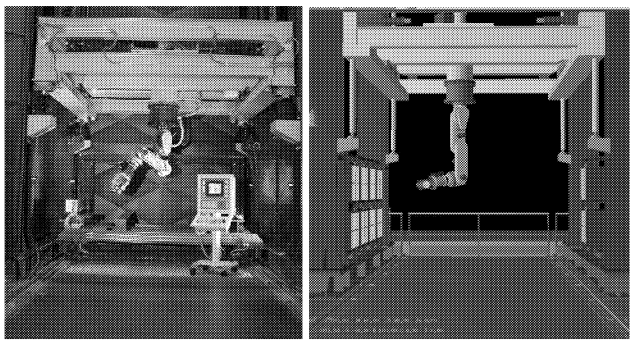


Figure 5. Simulation of a Heavy Redundant Robot

Application for education

It is important to remark that for teaching purposes, *VirtualRobot* is in use during more than two years in different courses imparted by DISA-UPV, both in faculties and schools in UPV (in the Faculty of Informatics and the School of Industrial Engineering) as well as PhD or international courses (for example, in Cuba). A book (Mellado 2003) with tutorial and exercises has been written and included in *VirtualRobot* suite. Exercises go from program analysis cases, where the student, according to his/her program execution time evaluates economical benefits for the company, to layout design and programming of a whole robotic cell.

The distance computation component, together with the sensor model component, has been used for practical lectures on the PhD course about robotics on the topic of collision detection. Students implement new components with distance computation and check the computational cost of their program when this component is loaded instead of the standard one. They can take the best of enveloping methods as explained in the theoretical lectures to speed up the distance computation problem. The most profitable fact is that both, *VRS* and the external application that control a laser range sensor scanner (as commented in last section), are inalterable for any component selected. Then the results are displayed in function of the component implemented by the student.

New applications for education will be commented in some of following sections.

Application for walking robot simulation

A walking robot is formed with a body and a set of legs. Each leg is an articulated robot with independent local control, while a global coordinated controller is in charge to decide the references for each leg according to the user order but restricted to some stability constraints. At any moment, the configuration of the legs will determine the location of the robot body.

After a research project working on walking robots, an external application was developed to run over *VRS* in order to simulate the behaviour of an insect like robot (Figure 6). Four threads are simulating the control action of leg controller, while the main application was processing walking orders from the user and computing body locations. Direct kinematics is used for a leg-walking step, while inverse kinematics is used to fix leg tips on the floor while the robot body is moving to achieve the walking step. With the help of this application, control algorithms can be tested in order to validate programs before checking in the real robot.

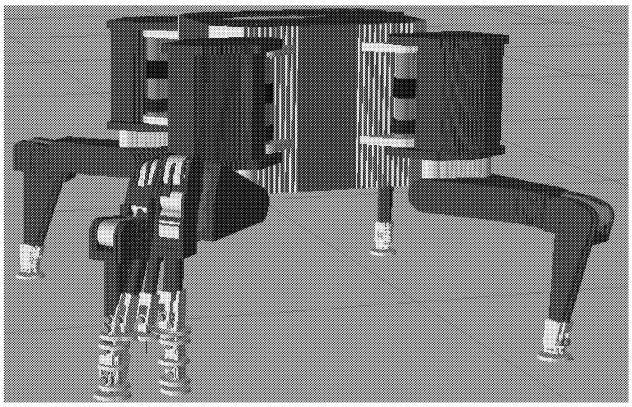


Figure 6. Simulation of a Walker Robot

Application for mobile robot simulation

In order to simulate mobile robots, special software in the form of a toolbox has been developed. *VirtualRobot* calls *craft* to any mobile robot that adapts to the following definition: "a craft is some kind of vehicle designed for navigation in water (rarely in other liquids), in the air or through space, which can move freely in its surrounding constrained to its kinematics and dynamics characteristics". The constraints can be inherent or temporal: when a craft moves on a surface such as a road, this can be because its design characteristics (for example, an automobile) or because its control status (for example, an airplane before taking off).
Usual cases of crafts can be classified on:

- Navigation in space: spacecrafts, rockets, …
- Navigation in air: airplanes, helicopters, …
- Navigation in water: submarines, …
- Navigation *on* water: ships, boats, yachts, ferries, …
- Navigation on ground: bicycles, tricycles, cars, …

The motion of a craft is obtained usually with a rotating device that produces movement in a linear direction. These devices that produce ahead movement are called ahead-drives. Helices and wheels are examples of ahead-drives. An ahead-drive can be active or passive. An active ahead-drive is one that its rotation produces a motion on the craft. On the other hand, in a passive ahead-drive is the motion of the craft that produces the rotation of the drive. That is, in a bicycle, the front wheel is a passive ahead-drive while the rear wheel is an active ahead-drive. The steering is obtained with a rotating device that produces turns on the craft. These devices that produce turning movements are called steering-drives. Flaps, helms and rudders are examples of steer-drives. Some times, both drives can be coupled, with an ahead-drive attached to a steer-drive, giving the steer-ahead-drive. The pair handlebar-wheel of a tricycle is an example. To model a craft, all these drive components must be modelled, specifying its particular motions. On the other side, the craft kinematics must be implemented in order to produce the motion correctly.

The craft toolbox is based mainly on a speed control thread that, each sample time, will call to a kinematics component. The kinematics component can be modified in order to be adapted for the specific craft to be simulated. It is the function of this component to locate properly the craft and all its drives. An external application must send speed commands to the craft's speed control thread according to user input. Two main applications of craft control are on use:

- Application to wheeled robots. A wheeled robot is formed with a body and a set of wheels. According to wheels distribution, there are different configurations for wheeled robots, but the most common ones are differential, bicycle, tricycle and car-like configurations (kinematics component for these types are included in *VirtualRobot*). Different applications for wheeled robot control have been implemented *over VRS*. Their use is very interesting for both, research and teaching. For example, works on kinematics control or lectures about robot navigation in mazes are imparted using these applications. Figure 7 shows an example of exercise where students must implement navigation strategies in order to pass a maze. On the other hand, some research projects for robot behaviour and map generation are on using *VirtualRobot*.

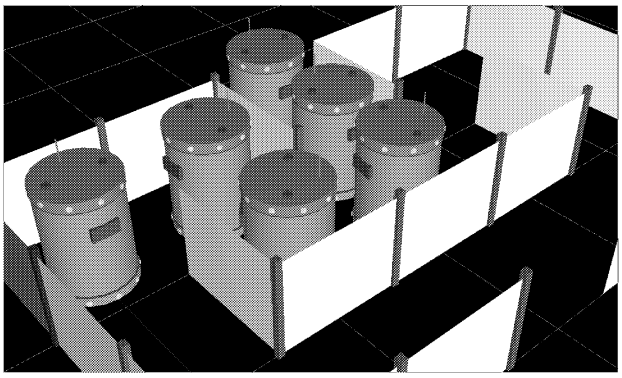


Figure 7. Simulation of a Wheeled Robot

- Application to underwater vehicles. An unmanned underwater vehicle is formed with a body and a set of actuators (thrusters and rudders). Possible kinematics configurations grow much more than for wheeled robots, but the working process is similar. Two main research works about underwater vehicles have been carried on using *VirtualRobot*: A dynamic trajectory control for Autonomous Underwater Vehicles (AUVs) and simulation of tele-manipulation tasks for the new category of underwater vehicles called Intervention Autonomous Underwater Vehicles (I-AUVs).

  In the first case, an external application for underwater vehicles simulation has been developed with optimum results. This application is composed of several modules that include the tasks of mission script interpreter, path generation, control, and dynamic model vehicle simulation. The vehicle control is implemented using an independent module, giving to the user the possibility to develop and test different control architectures and strategies. For the dynamics, the most common model for submarines (Fossen 1994) has been selected. The vehicle's dynamic and hydrodynamic parameters are defined using structured ASCII files that give a high

flexibility for modelling, testing and designing different vehicle configurations. Simulation results are sent to *VRS* for graphical representation. Figure 8 shows a simulation generated by this application with the reference path in dotted line and the simulated path in continuous line. The difference between them is due, mainly, to sea currents, control deficiencies, and limitations in the thrusters' dynamics.

In the second case, *VirtualRobot* is used to simulate the tele-manipulation task carried on by an I-AUV (Figure 9), focusing in the manipulator trajectory correction, necessary for compensating the vehicle's residual movement after the docking maneuver (Catret et al 2002).

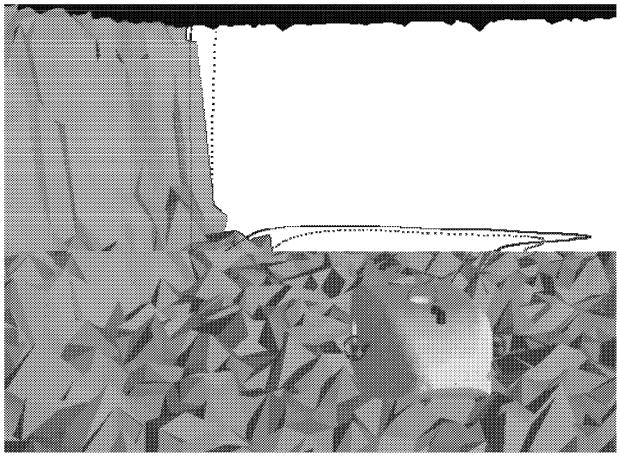


Figure 8. Simulation of an Underwater Vehicle

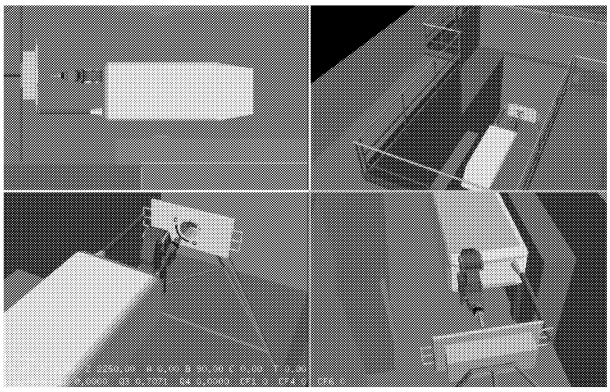


Figure 9. Simulation of an Underwater Teleoperation Task

Application for sensor simulation

VirtualRobot is not only applied to robot simulation, but also to sensor simulation, mainly range sensors because of their wide use on robotics. Two examples will illustrate its potentiality:

- Sensor based applications. *VirtualRobot* has been used as a valuable product for sensor based 3D-mapping validation on robotics, as published in (Catret et al. 2001). In this work, for example, 3D-maps realistic and smart graphics representation can be easily generated using simple functions, in such a way that it is possible to implement sensor models, signal filters, 3D-mapping methods or data fusion. *VRS* is used to display the result of processing different filtering and map generation processes (for example, 2D and 3D probabilistic methods based on Bayes' Theorem) for the data captured with an ultrasonic sensor attached on a manipulator (Figure 10). Furthermore, the specific physical sensor and robot are transparent to the user when deciding to take the best of the hierarchical software structure offered to manage this kind of systems. A simple case implementation code is analyzed and the results of a more complex example are discussed in the mentioned paper. New teaching opportunities have been raised when different sensor models are implemented and their output verified and compared with this application.

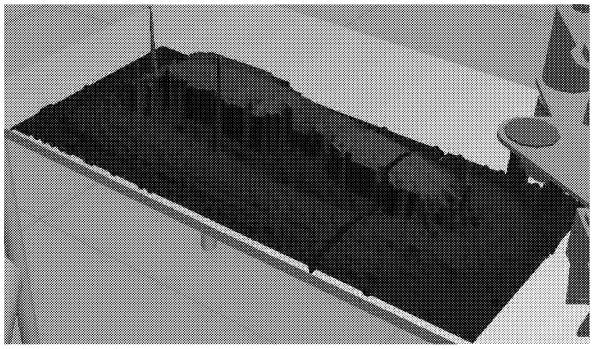


Figure 10. US Sensor Scanning Simulation

- Laser Range Finder (LRF) simulation. A LRF emits a laser beam and detects the echoed of the emitted beam. Measuring the time of fly, it is possible to calculate the distance from the laser to the object detected. 3D reconstruction allows constructing a 3D model as complete and as accurate as possible, from a real world scene, for which a priori information is not available. For doing this, it is necessary to locate sensors at several locations in the environment, until all surfaces will be visible (Sequeira et al. 1999a; Sequeira et al. 1999b). *VirtualRobot* permits to build several types of lasers 1D, 2D and 3D as well as modelling their behaviour (Figure 11). In addition, *VirtualRobot* is used to test the perception planning algorithms that consent to calculate the sequence of locations where to place the laser sensor for the next view and where to look at to resolve the occlusions in the scene.

Other application when this technology can be used is the surveillance. Taking a profile of the area to be under surveillance, it is possible to detect some changes and generate an alarm, comparing the current laser profile with the reference profile. *VirtualRobot* can be used by the inspectors to decide when and where to situate the sensors to cover all the area to be under surveillance.

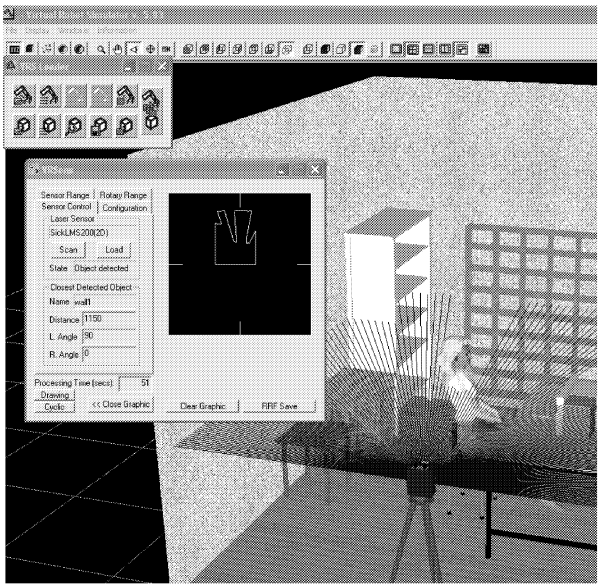


Figure 11. Laser Range sensor Scanning Simulation

CONCLUSIONS

The great possibilities of *VirtualRobot* simulation software for educational, research and industrial purposes in robotics have been presented in this paper through different applications. The main implementation basis has been deeply exposed to show its great advantages. The versatility and flexibility of *VirtualRobot* is verified when used on different mechanical platforms, as explained for different works, such as manipulators, mobile robots (walking, wheeled, and underwater robots) and sensor systems. The reader must notice that all the figures that illustrate the paper have been obtained from the simulation application examples developed on *VirtualRobot* and therefore have the same user interface.

ACKNOWLEDGEMENTS

This work has been partially funded by FEDER-CICYT project DPI2002-04434-C04-04. The authors also want to thank all the students that during last years have also participated, with more or less relevance, on *VirtualRobot* development. Some of the results presented in this paper are from projects of the units *Non Proliferation and Nuclear Safeguard* and *Requirements, Analysis and Verification* of IPSC (JRC).

REFERENCES

Catret, J.V.; Mellado, M.; Puig, D. 2001. "Open Graphics System for 3D-Mapping Validation on Robotics". *8th IEEE Int. Conf. on Electronics, Circuits and Systems*¸ Valetta Malta.

Catret, J.V.; Grosset D.; Papadopoulos C.; Andritsos F. 2002. "Trajectory correction of an AUV-mounted Underwater Robotic Manipulator". *4th International Workshop on Computer Science and Inforemation Technologies (CSIT).*

Chen, X; Marco, D; Smith, S; An, E; Ganesan, K.; Healey, T. 1997. "6 DOF Nonlinear AUV Simulation Toolbox". *IEEE*, Oceans 97, Halifax.

Fossen, T., 1994. "Guidance and Control of Ocean Vehicles". *John Wiley & Sons Ltd.*

Freund, E.; Rokossa, D.; Rossman, J. 1999. "Intuitive Off-line Programming of Industrial Robots Using VR-Techniques", *15th ISPE/IEE Int. Conf. on CAD/CAM, Robotics and Factories of the Future*, Águas de Lindóia, SP, Brazil.

Hale R.D.; Rokonuzzaman M.; Gosine R.G. 1999. "Control of Mobile Robots in Unstructured Environments using Discrete Event Modeling". *The Int. Society for Optical Engineering.*

Howie, P. 1984. "Graphic Simulation for Off-Line Robot Programming". *Robotics Today*. February. ISSN: 0193-6913.

Kimoto, K.; Yuta S.I. 1995. "Autonomous Mobile Robot Simulator: a Programming Tool for Sensor-based Behavior". *Autonomous Robots*. IS: 0929-5593.

Matsumoto, Y.; Miyazaki, T.; Inaba, M.; Inoue, H. 1999. "View Simulation System: A Mobile Robot Simulator using VR Technology". *IEEE Int. Conf. on Intelligent Robots and Systems*. Piscataway, NJ, USA.

Mellado, M.; Catret, J.V.; Puig, D.; Vendrell, E. 2001a. "Sistema Remoto de Programación, Simulación y Monitorización de Robots". *Simposio de Ingeniería Eléctrica (SIE 2001)*, Santa Clara, Cuba.

Mellado, M.; Catret, J.V.; Puig, D. 2001b. "Dispositivos de Entrada de Datos 3D en Tele-Robótica". *2º Congreso Int. Interacción Persona-Ordenador*, Salamanca, Spain.

Mellado, M.; Sánchez, A.J.; Vendrell, E.; Zotovic, R. 2002. "Practicas de Programación de Robots". *Editorial Universidad Politécnica de Valencia.* ISBN: 8497052455.

Mellado, M. 2003. "Simulación en robótica mediante *VirtualRobot*". *Editorial Universidad Politécnica de Valencia.* Ref: 2003.443.

Parker, J. 1985. "Simulating a Robot Arm using Graphics and Animation". *AI, Graphics and Simulation* Published by Soc for Computer Simulation, La Jolla, CA, USA.

Pérez, M. and Schilling, K. 2001."Virtual Reality for Tele-Education Experiments with Remote Mobile Robot Hardware". *Workshop on Internet Based Control Education.*

Puig, D.; Mellado, M.; Catret, J.V. and Ruiz, E. 2001. "Robot Kinematics Development Environment: Application to a Configurable Redundant Manipulator for Heavy Robotics". *5th World WSES/IEEE Conf. on Circuits, Systems, Communications & Computers*, Creta, Greece.

Ruiz, E. 1999. "GENERIS: The EC- JRC Generalised Software Control System for Industrial Robots". *Int. Journal of Industrial Robot*, Vol. 26, No. 1.

Sequeira V.; Ng K.; Wolfart E.; Gonçalves J.G.M.; Hogg D.C. 1999a. "Automated 3D reconstruction of interiors with multiple scan-views". *SPIE Proc. Vol. 3641-Videometrics VI, SPIE-Photonics West - Electronic Imaging'99 Int. Symp., pp. 106-117*, San Jose (CA), USA.

Sequeira V.; Ng K.; Wolfart E.; Gonçalves J.G.M.; Hogg D.C. 1999b. "Automated Reconstruction of 3D Models from Real Environments", *ISPRS Journal of Photogrammetry and Remote Sensing (Elsevier), vol. 54, pp. 1-22*, February.

Stauffer, R.1984. "Robot System Simulation". *Robotics Today*. June. ISSN: 0193-6913

Yoshida K. 1995. "Experimental Study on the Dynamics and Control of a Space Robot with Experimental Free-Floating Robot Satellite (EFFORTS) Simulators". *Advanced Robotics*. ISSN: 0169-1864.

Yang, L.; Yang, X.; He, K.; Guo, M.; Zhang, B. 1997. "Research on Mobile Robot Simulation & Visualization". *IEEE Int. Conf. on Systems, Man and Cybernetics*. Piscataway, NJ, USA.

Zotovic, R.; Mellado, M.; Jornet, V.; Catret, J.V.; Puig, D. 2001. "Diseño, Implementación y Control de un Sistema Háptico con Realimentación Sensorial en Tele-Robótica". *2º Cong. Int. Interacción Persona-Ordenador*, Salamanca, Spain.

SLIDING MODE CONTROL APPLIED TO MOBILE ROBOT: CONTINUOUS-TIME AND DISCRETE –TIME CASE

M. Hamerlain and T. Alalouche
Laboratoire de Robotique et d'Intelligence Artificielle, CDTA
Houch OUKIL, Baba-HassenBP. 17 Baba-Hassen, Alger, Algérie
E-mail : mhamerlain@cdta.dz

ABSTRACT

In this work a sliding mode controller solution is proposed for tracking problems associated to well known types of mobile robots. The robustness property of the sliding mode controller for the continuous-time systems can thus be extended to the discrete-time systems under others conditions for existence sliding modes.

Keywords: Simulation, discrete-time and continuous-time sliding mode controller, mobile robot.

INTRODUCTION

Sliding mode control systems, also known as Variable Structure Control Systems (VSCS), are characterized by a discontinuous feedback control law which switches the structure of the system during the evolution of the state vector in order to maintain the state trajectories in a predefined subspace (Drakunov and Utkin 1989).

Conventional sliding mode control designed on the basis of a continuous-time system is known to be robust to the plant uncertainties. Its implementation by a digital computer, however, requires a certain sampling interval which brings not only chattering but also possible instability.

Discrete-time sliding mode controllers have usually been developed using state space models (Chan 1997), (Furuta 1993), (Surpturk et al. 1987), (Sira-ramirez 1991), (Spurgeon 1992).

Its use allows the selection of the sliding surface to be similar to that for the continuous-time case.

The objective of this paper is a performance comparison of two algorithms continuous case and Discrete case of sliding mode control applied to mobile robot.

The organization of this article is as follows. Section 2 reviews continuous-time sliding mode control, and section 3 gives a stability analysis for discrete-time sliding mode control with a sufficient condition for existence sliding mode. Finally section IV contains a conclusion.

SLIDING MODE CONTROL FOR CONTINUOUS SYSTEM:

We consider a Wheeled Mobile Robot of type (3.0). The posture Dynamical Model for this mobile robot is given by the following six equations (Chacal Sira-Ramirez 1994):

$$\begin{cases} \dot{x} = \eta_1 \cos\theta - \eta_2 \sin\theta \\ \dot{y} = \eta_1 \sin\theta + \eta_2 \cos\theta \\ \dot{\theta} = \eta_3 \\ \dot{\eta}_1 = u_1 \\ \dot{\eta}_2 = u_2 \\ \dot{\eta}_3 = u_3 \end{cases} \quad (1)$$

Where : x,y are the WMR position and θ is its orientation ; η_1,η_2 and η_3 are velocities and u_1,u_2,u_3 are the control inputs which are acceleration-homogeneous.

From (1), we obtain η_1,η_2,η_3 and u_1,u_2,u_3 as follows:

$$\eta_1 = \frac{\dot{x}}{\cos\theta} + \tan\theta(\dot{y}\cos\theta - \dot{x}\sin\theta)$$

$$\eta_2 = \dot{y}\cos\theta - \dot{x}\sin\theta$$

$$\eta_3 = \dot{\theta}$$

$$u_1 = \frac{\ddot{x}\cos\theta + \dot{x}\dot{\theta}\sin\theta + \dot{\theta}(\dot{y}\cos\theta - \dot{x}\sin\theta)}{\cos^2\theta} +$$

$$\tan\theta(\ddot{y}\cos\theta - \dot{y}\dot{\theta}\sin\theta - \ddot{x}\sin\theta - \dot{x}\dot{\theta}\cos\theta)$$

$$u_2 = \ddot{y}\cos\theta - \dot{y}\dot{\theta}\sin\theta - \ddot{x}\sin\theta - \dot{x}\dot{\theta}\cos\theta$$

$$u_3 = \ddot{\theta} \quad (2)$$

The control objective is to make the three posture state variable x, y, θ tend to their desired values:

$$e_1 = x - x_d \; ; \quad e_2 = y - y_d \; ; \quad e_3 = \theta - \theta_d$$

We define the sliding surfaces for $i = 1,2,3$ as:

$$S_i = \dot{e}_i + \lambda_i e_i \quad \text{for} \quad i = 1,2,3$$

Following the sliding mode reaching law approach, the dynamics of the sliding surfaces are assigned by defining their variations as follows:

$$\ddot{e}_i + \lambda_i \dot{e}_i + W_i sign S_i = 0 \qquad (3)$$

From the expression (5), the existence condition of the sliding mode must be satisfied and is equivalently as: $\dot{S}_i = -W_i sign S_i$ (4)

The controls inputs law $v_i(k)$ become:

$$v_1 = \frac{(\ddot{x}_d - \lambda_1(\dot{x} - \dot{x}_d) - W_1 sign(S_1))\cos\theta}{\cos^2\theta} + \frac{\dot{x}\dot{\theta}\sin\theta + \dot{\theta}(\dot{y}\cos\theta - \dot{x}\sin\theta)}{\cos^2\theta} + \tan\theta((\ddot{y}_d - \lambda_2(\dot{y} - \dot{y}_d) - W_2 sign(S_2))\cos\theta - \dot{y}\dot{\theta}\sin\theta - (\ddot{x}_d - \lambda_1(\dot{x} - \dot{x}_d) - W_1 sign(S_1))\sin\theta - \dot{x}\dot{\theta}\cos\theta)$$

$$v_2 = (\ddot{y}_d - \lambda_2(\dot{y} - \dot{y}_d) - W_2 sign(S_2))\cos\theta - \dot{y}\dot{\theta}\sin\theta - (\ddot{x}_d - \lambda_1(\dot{x} - \dot{x}_d) - W_1 sign(S_1))\sin\theta - \dot{x}\dot{\theta}\cos\theta$$

$$v_3 = \ddot{\theta}_d - \lambda_3(\dot{\theta} - \dot{\theta}_d) - W_3 sign(S_3)$$

SLIDING MODE CONTROL FOR DISCRETE SYSTEM

By introducing a sampling process in the control algorithm (1) can be transformed to its discrete equivalent taking into account the delay of a sampling period. The discretized version of (1) becomes the following discrete-time system:

$$\begin{cases} x(k+1) = x(k) + T[\eta_1(k)\cos(\theta(k)) - \eta_2(k)\sin(\theta(k))] \\ y(k+1) = y(k) + T[\eta_1(k)\sin(\theta(k)) + \eta_2(k)\cos(\theta(k))] \\ \theta(k+1) = \theta(k) + T\eta_3(k) \\ \eta_1(k+1) = \eta_1(k) + Tu_1(k) \\ \eta_2(k+1) = \eta_2(k) + Tu_2(k) \\ \eta_3(k+1) = \eta_3(k) + Tu_3(k) \end{cases} \qquad (5)$$

After dynamically feedback linearizing the robot dynamical model, the Discrete Variable Structure Control (DVSC) reaching law is applied for a robust stabilization.

A discrete-time version of the dynamical model using the forward difference discretizing formula, is:

$$x(k+1) = x(k) + T[\eta_1(k)\cos(\theta(k)) - \eta_2(k)\sin(\theta(k))]$$

$$x(k+2) = x(k+1) + T[\eta_1(k+1)\cos(\theta(k+1)) - \eta_2(k+1)\sin(\theta(k+1))]$$

Substituting into (5)

$$x(k+2) = x(k+1) + T[(\eta_1(k) + Tu_1(k))\cos(\theta(k+1)) - (\eta_2(k) + Tu_2(k))\sin(\theta(k+1))]$$

$$y(k+2) = y(k+1) + T[(\eta_1(k) + Tu_1(k))\sin(\theta(k+1)) + (\eta_2(k) + Tu_2(k))\cos(\theta(k+1))]$$

$$\theta(k+2) = \theta(k+1) + T(\eta_3(k) + Tu_3(k))$$

In Vectorized form, we obtain:

$$\begin{pmatrix} x(k+2) \\ y(k+2) \\ \theta(k+2) \end{pmatrix} = GU(k) + H \qquad (6)$$

with $U(k) = \begin{pmatrix} u_1(k) \\ u_2(k) \\ u_3(k) \end{pmatrix}$ and T=0.01 s

Now, we define $V(k) = (v_1(k), v_2(k), v_3(k))^T$ as the auxiliary input-variables that set the desired linear dynamics of the three system-output variables as follows:

$$\begin{cases} x(k+2) = v_1(k) \\ y(k+2) = v_2(k) \\ \theta(k+2) = v_3(k) \end{cases} \qquad (7)$$

From (7), we obtain:

$$U(k) = G^{-1}(-H + V(k)) \qquad (8)$$

Then, the nonlinear system, as linearized and decoupled, can be written into three sub-systems, as follows:

$$\begin{aligned} X_1(k+1) &= A_1 X_1(k) + b_1 v_1(k) \\ X_2(k+1) &= A_2 X_2(k) + b_2 v_2(k) \\ X_3(k+1) &= A_3 X_3(k) + b_3 v_3(k) \end{aligned} \qquad (9)$$

with: $A_1 = A_2 = A_3 = A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$,

$$B_1 = B_2 = B_3 = b = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$X_1(k) = X(k) = \begin{pmatrix} x(k) \\ x(k+1) \end{pmatrix},$$

$$X_2(k) = Y(k) = \begin{pmatrix} y(k) \\ y(k+1) \end{pmatrix},$$

$$X_3(k) = \Theta(k) = \begin{pmatrix} \theta(k) \\ \theta(k+1) \end{pmatrix}$$

After this linearization, let us now define the sliding surfaces as:

$$S_i(k) = c_i^T E_i(k) \qquad (10)$$

Where c_i are the surfaces gains and

$$E_i(k) = X_i(k) - X_{désiré}(k), \quad i = 1,2,3$$

For the DVSC, an equivalent form of the reaching law is:

$$\begin{aligned} &S_i(k+1) - S_i(k) = -q_i T S_i(k) - w_i sign(S_i(k+1)) \\ &= c_i^T E_i(k+1) - c_i^T E_i(k) \end{aligned} \qquad (11)$$

with: $q_i, w_i > 0$ and $1 - q_i T > 0$.

The control input laws $v_i(k)$ are derived as:

$$\begin{aligned} &v_i(k) = -(c_i^T b)^{-1}(c_i^T A X_i(k) - c_i^T X_{désiré}(k+1) - \\ &(1 - q_i T) c_i^T E_i(k) + w_i T sign(c_i^T E_i(k)) \end{aligned} \qquad (12)$$

It is clear that the condition (4) which assures the sliding motion on the *i*th hyperplane in continuous systems is no longer applicable in discrete-time systems.

This condition is necessary but not sufficient for the existence of a quasi-sliding motion. Indeed, it does not assure any convergence of the state trajectories onto the *i*th hyperplane and may result in an increasing amplitude chatter of the state trajectories around the *i*th hyperplane, which means instability. Thus, a discrete-time sliding mode condition must be imposed. A necessary and sufficient condition will be imposed here assuring both sliding motion and convergence onto the *i*th hyperplane. This condition may be stated as :

$$|s_i(k+1)| < |s_i(k)| \qquad (13)$$

CONCLUSION

This paper has presented a sliding mode controller solution for tracking problems applied to mobile robots. The robustness property of the sliding mode controller for the continuous-time systems can thus be extended to the discrete-time systems.

REFERENCES

Chacal J. A. and H. Sira-Ramirez, 1994, "On the sliding mode control of Wheeled mobile robots", in Proc. IEEE Int. Conf. Systt. Man. Cybern, Sept. 1994, pp. 1938-1943.

Chan C. Y., 1997, "Discrete-time adaptive sliding mode control of a linear system in state space form". Int. J. Control, 1997, Vol. 67, N° 6, 859-868.

Drakunov, S.V, and Utkin, V.I., 1989, On discrete-time sliding modes. Proceedings of IFAC, Conference on Nonlinear Control Systems Designe, Capri, Italy, pp. 273-277.

Furuta, K. , 1993, "Sliding mode control of a discrete system". Systems and Control Letters, 14, 145-152. Proceeding of the American Control Conference, San Francisco, California, U.S.A, pp.980-984

Sarpturk, S., Y. Istefanopulos and O. Kaynak, 1987. " On the stability of discrete-time sliding mode control systems. IEEE Transactions on Automatic Control, 10, 930-932.

Sira-Ramirez, H., 1991,"Non-linear discrete variable structure systems in quasi-sliding mode". Int. Journal of Control, 54,
1171-1187.

Spurgeon, K. S., 1992, "Hyperplane design techniques for discrete-time variable structure control systems", Int. Journal of Control, 55, 445-456.

BIOGRAPHY

Mustapha Hamerlain received the Doctorate degree in 1993 from National Institute of Applied Sciences of Toulouse (France). He is currently with the Advanced Technologies and Development Center (Algiers). His research interests are sliding mode control, robust control and visual control.

Alalouche Tahar received the DEA (Diplôme des Etudes Approfondies) in 2003 from National Polytechnic School of Algiers (Algeria). Currently he is PHD student at SUPELEC Paris (France). His research concentrates on Mobile Robot Control.

SIMULATION TOOLS IN DISTRIBUTED NETWORKS

ON USING AN EMULATIVE MIDDLEWARE TO MODEL WIRELESS NETWORKS: SIMULATION RESULTS AND VALIDATION

Stefano Cacciaguerra, Marco Roccetti, Vittorio Ghini, Stefano Ferretti
Department of Computer Science, University of Bologna
Mura A. Zamboni 7, 40127 Bologna, Italy
Phone: +39 0512094503 - Fax : +39 0512094510
E-Mail: { scacciag, roccetti, ghini, sferrett}@cs.unibo.it

KEYWORDS
End-to-End Network Simulation, Emulation, Performance Evaluation, Wireless Simulation Model, Test Analysis.

ABSTRACT

Setting up and using realistic wide area testbeds for developing and testing large-scaled distributed applications and communication mechanisms may be very expensive. Therefore, designers use simulative technologies to obtain elastic and reproducible frameworks where those experiments may be safely carried out in local lab-based environments. Conventional simulators are based on the idea that the model of the real system must be completely simulated. In contrast with this idea, most recent simulation technologies say that not all entities should be simulated, but some should be real (Fall, 1999). That is, the simulation experiments must not be confined to isolated simulation worlds but have to embrace also emulation scenarios with real operational network devices. By exploiting this idea, we have designed and developed a simulative platform for networked applications (called Interceptor), where an emulation interface has been developed: (i) to integrate into the simulation real network communication traffic coming from different networked end-systems, and (ii) to incorporate into the simulation environment entire communication protocol layers without the need of specifying them into the simulation model. In this paper we present the architecture of Interceptor and discuss some relevant examples of wireless simulation modeling using our emulation-based approach.

INTRODUCTION

The development of modern network technologies has allowed the proliferation of distributed applications and communication mechanisms that allow an increasing number of users, geographically dispersed throughout the world, to access to large-scaled networked infrastructures through the use of a plethora of different devices, such as, for example: networked computers, smart phones and personal digital assistants. Those distributed applications are gradually evolving from the traditional client-server model to new architectural approaches, based on the idea of sharing among groups of peers data resources, computational power, and network bandwidth. Such applications are typically characterized by the possibility of replicating data and distributing computations needed to perform complex cooperative tasks, such as supporting real-time multimedia interactions and sharing collaborative digital tools. Relevant examples of emerging services provided by those applications are: videotelephony, teleconferencing, instant multimedia messaging, teleworking, telemedicine, real time entertainment, interactive games and infotainment.

Setting up and using realistic wide area testbeds for developing and testing those large-scaled distributed applications may be very expensive. Therefore, designers use simulative technologies to obtain elastic and reproducible frameworks where those experiments may be safely carried out in local lab-based environments. Conventional simulators are based on the idea that the model of the real system must be completely simulated. In contrast with this idea, most recent simulation technologies (e.g., the NS network simulator (Fall, 1999)) say that not all entities should be simulated, but some should be real. That is, the simulation experiments must not be confined to isolated simulation worlds but have to embrace also emulation scenarios with real operational network devices. In this context, we have designed and developed a coarse-grained network simulative platform (called Interceptor) which incorporates an emulation interface specifically developed to integrate into the simulation model, real network communication traffic coming from different end-systems, such as clients, servers, peers and proxies. The main novelty of our approach amounts to the fact that we have adopted a simulative technology which interposes a coarse-grained simulator as a centralized intermediate node among real sources of traffic networks. Using this approach, it is possible to support the real execution of distributed applications on real network nodes, while simulating at a higher abstraction level the end-to-end behavior of network traffic (Roccetti et al., 2002a). One of the prominent advantages of this approach is that the performance of simulation is favored without sacrificing the accuracy of simulation modeling. In this paper we discuss on an example of simulation modeling for wireless communications with Interceptor. Based on the idea that the end-to-end traffic behavior is simulated within Interceptor at a higher abstraction level, several wireless technologies, e.g. GPRS, UMTS and Wi-Fi have been modeled by conducting a regression analysis on real data coming from on-the-field trials. To reproduce a variety of different communication environments, each developed wireless model has been built based on the possibility to tune specific communication attributes (e.g., user's speed,

external traffic in the cell). The experimental study, we have carried out, shows that our developed simulative models are very effective for the testing of wireless distributed applications in a lab-based environment. Additionally, we conducted a complete validation analysis based on the standard statistical theory that has confirmed the correctness of our wireless models (Banks et al., 2001).
The reminder of this paper is organized as follows. In Section 2, we introduce the simulative architecture which is at the basis of Interceptor. In Section 3, we report on our wireless modeling exercises, while Section 4 illustrates our validation tests. Finally, Section 5 provides some concluding remarks.

INTERCEPTOR: SIMULATIVE ARCHITECTURE

The adopted models for simulating network traffic communications may follow two different schemes: either a fine-grained approach or a coarse-grained approach. In the fine-grained approach (Breslau et al., 2000), the simulative model of network traffic communications incorporates meticulously the specification of each protocol layer. According to this approach, separate simulation models may be defined, e.g., for the data link level in point-to-point links and LANs, and for the network level in IP-based routers and hosts. Besides the composition at the simulation abstraction level, the main problem of this approach amounts to scalability problems derived from the need of integrating the simulation model with the traffic coming from hundreds of real network devices. In this case, when the simulator collects such external traffic and passes it up to the simulation model, performance degradation may occur. In simple words, the more the accuracy of simulation modeling, the less the simulation performance.

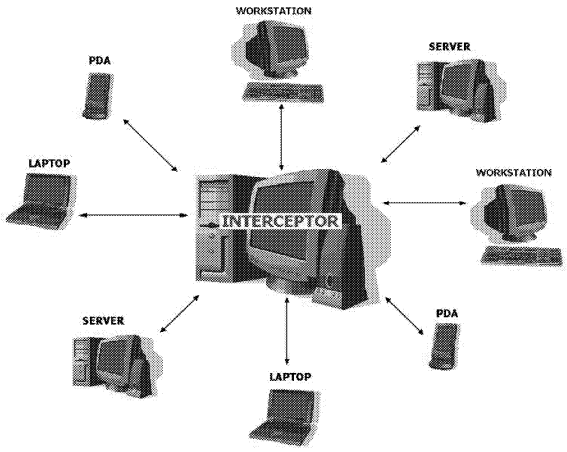


Figure 1. Interceptor: Intermediate Simulation Node

Alternatively, the latter (i.e., the coarse-grained) approach models large-scaled network settings sacrificing simulation details, while maintaining the possibility of integrating external traffic and real devices. This approach prefers the performance of simulation to the accuracy of the model, without sacrificing the validity of this one. Interceptor, the simulation tool we have developed, adopts this latter approach. It implements a centralized intermediate simulation node which is able to intercept IP packets exchanged between a couple of real network objects, while simulating packet delays and losses according to a predefined comprehensive end-to-end traffic model that abstracts from complex simulation details. In essence, Interceptor allows simulationists to use real network IP-based devices (see Figure 1) and real protocol implementations, while maintaining unchanged the network communication interface. Besides, it allows to confine in a lab-based environment such real network objects, while simulating a network traffic setting with appropriate parameters. Finally, it allows to change and update the simulation model without affecting the emulation interface.
After the explanation about what Interceptor is and which its advantages are, we want to illustrate how it works. Interceptor is composed by two different parts: an emulation interface (EI) and a simulative environment (SE) (see Figure 2). The EI allows the simulation engineer to integrate real computing devices into the simulative environment. In particular, the EI accommodates network devices interconnected by means of a multi-homed host, running the Linux operating system. Interceptor plays the role of a gateway for each different internetworked device.

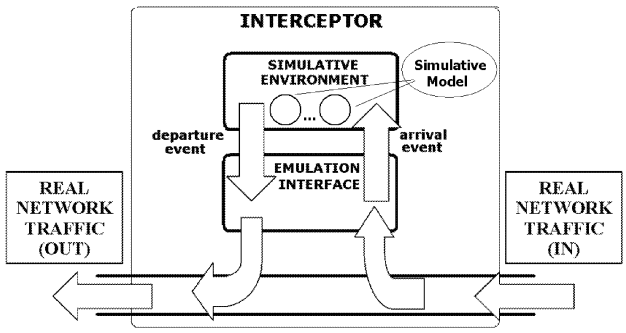


Figure 2. Interceptor: Simulative Architecture

Since Interceptor must act as the only intermediate simulation node among all the different workstations, sophisticated net masking techniques have been implemented to guarantee the correct behavior of packet flows through the simulation. In particular, to enforce each workstation to communicate only through the default gateway, the net mask of such workstation must be configured so that there is a one-to-one mapping between it and the corresponding subnet. In simple words, each subnet holds a single workstation. Due to this software configuration of the IP layer, each IP datagram sent from a given workstation towards its peer passes through the default gateway that, finally, forwards it to the destination. When a datagram arrives to the default gateway, Interceptor delays or discards such datagram according to the predefined simulation model. It is worth mentioning that the communication traffic, entering in the simulation environment, must exit under real-time constraints. These real-time constraints are managed by the Simulative Model (SM). In essence, when an IP packet arrives to the default gateway, the EI passes it to the SE and marks its arrival time (arrival event). The SE inserts the IP packet in a waiting queue and computes the simulation time the packet

must wait in the queue, according to the rules specified in a predefined SM. Upon completion of this waiting time, SE sends IP packet to EI (departure event). EI is in charge of sending out the delayed packet towards its external networked destination. As a final consideration, it is worth reporting on how we have modeled within Interceptor large numbers of users running simultaneously the same application. As replicating several machines, each running a copy of the same application, is not practical we have resorted to the approach suggested in (Barford et al., 1998). In essence, following this approach, we have developed a realistic workload generator which mimics a set of real users running the same application on the same machines (Cardellini et al., 2001). This permits to Interceptor to simulate up to several tens of users who simultaneously run the same application on the same machine, in emulation mode.

WIRELESS SIMULATION MODELING

We describe an example of wireless communication modeling using Interceptor. In particular, we show how we have obtained from a set of traces of real wireless data, the correspondent wireless SMs to be embodied in the SE (Roccetti et al., 2002b; Salomoni, 2003). To this end, we have exploited three different wireless infrastructures, namely Wi-Fi, GPRS and UMTS. Specific attributes of each different technology were varied to build the correspondent SM. In essence, the model may be thought of as a black-box (see Figure 3), where the input variables I={IN.1,…,IN.k} represent these real attributes, while the output O={OUT} is computed on the basis of a regressive function f(I)=O that accounts for the time needed to transmit/download IP packets over a wireless link (i.e., Download Time).

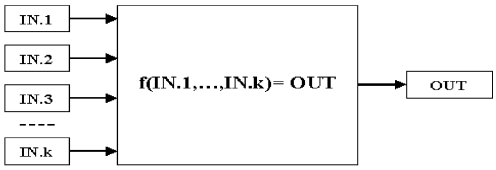


Figure 3. Black-Box Model

To carry out our regressive analysis we have tried with several different functions, but the best ones have turned out to be those shown in the quality test reported in Table 1. For the sake of completeness, it is worth mentioning that the input attributes of each wireless infrastructure were: i) the size of the transmitted file (expressed in KB), ii) the speed at which users move throughout the cell (expressed in Km/h), iii) the additional traffic in the cell (expressed via Erlang values), iv) the coverage quality (good: device under coverage; bad: handover situations). To conclude this Section, we report in Figure 4 some of the wireless modeling results we have obtained for the wireless architectures we have examined. The simulative models for the Wi-Fi, GPRS and UMTS connections are presented in Figure 4 where the time needed to transmit/download test files (Download Time expressed in seconds) is the output value.

Table 1. Quality Test for Regressive Functions

Architecture	Function	R	R^2	Adj R^2	Std. Err. of Estimate
GPRS	$f = a \cdot x + b \cdot y$	0,9971	0,9942	0,9934	19,0794
UMTS	$f = a \cdot x + b \cdot y$	0,9999	0,9999	0,9999	1,3252
Wi-Fi	$f = c + b \cdot x + a \cdot x^2$	1,0000	1,0000	-	-

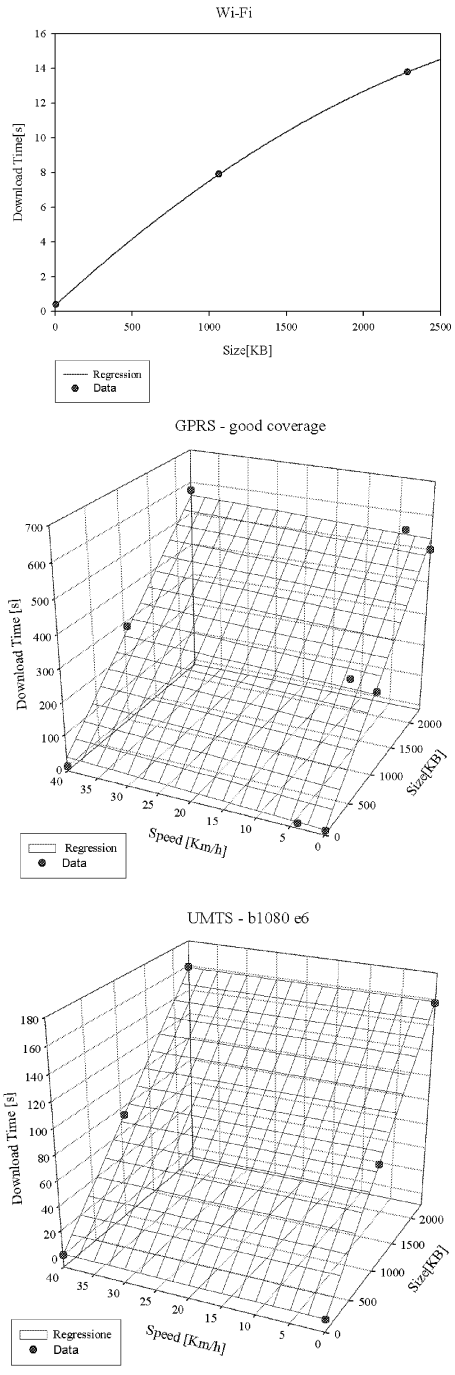


Figure 4. Wireless Modeling Results: Wi-Fi (top), GPRS (middle), UMTS (bottom)

For a better understanding of these results it is worth reporting the following considerations on the traces of real data we used to build our wireless models of Figure 4. The transmitted files used for our experiments were stored on a web server with the following characteristics: it was a Pentium 3 machine with 1 GHz, 512 MB RAM, and Windows 2000 Advanced Server. On the client side, a PDA was exploited with the following characteristics: it was an IPAQ 3850 compact with Microsoft Pocket PC. To collect real traces over the GPRS infrastructure (Bettstetter et al., 1999) (Kalden et al., 2000) the IPAQ was equipped with a GPRS expansion pack with a card of a GPRS Italian provider. Since at the time of our measurements the UMTS infrastructure was not available, an UMTS network emulator was used to obtain UMTS based-traces. Finally, in the Wi-Fi case, a PCMCIA expansion pack with IEEE a 802.11b card for the IPAQ and a LINKSYS access point, were adopted. We have assumed that in this case users were still and their number was quite low. This justifies the fact that the first plot of Figure 4 is depicted on a plane.

VALIDATING THE WIRELESS MODELS

In this final Section, we report on the validation strategy we have adopted to confirm our wireless models. In particular, to control the viability of our wireless models, we have followed two alternative schemes. According to the first scheme, we have contrasted the real data, we have collected with on-the-field trails, with the results generated by our wireless models. Based on the second scheme, instead, we have evaluated the probabilities of rejecting a valid model (Type I error) and accepting an invalid model (Type II error) (Banks et al., 2001). As to the first scheme, we have obtained an average difference of approx. 5% between real and simulated data. Examples of this validation activity are reported in Figure 5, for respectively the Wi-Fi, GPRS and UMTS architectures.

According to the second scheme, and based on the standard theory for the calibration of the simulative models, we have formally conducted a statistical test of the null hypothesis:

$$H_0 : E(T) = \mu_0 \quad \text{versus} \quad H_1 : E(T) \neq \mu_0 \qquad (1)$$

where the average delay of the system response (μ_0) is compared with the average result provided by the wireless models ($E(T)$). We carried out the test hypothesis H_0 with a probability of rejecting H_0, when H_0 is true, equal to α = 0.01 and a sample of n = 30 experiments. Hence, we have computed the test statistics:

$$t_0 = \frac{E(T) - \mu_0}{S/\sqrt{n}} \qquad (2)$$

where μ_0 is the specified value in the null hypothesis H_0, $E(T)$ is the average delay of the model response and S is the standard deviation of T.

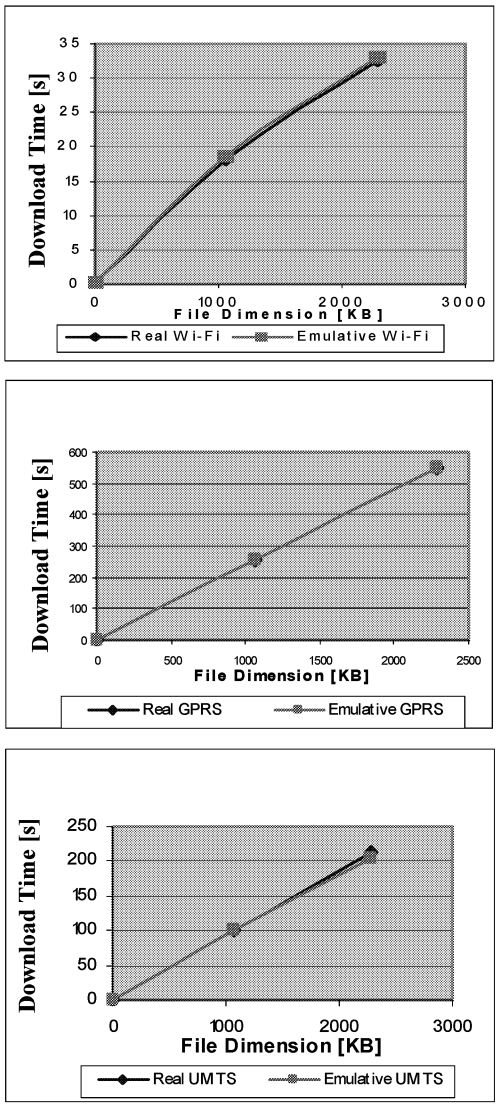


Figure 5. Validation:
Wi-Fi (top), GPRS (middle), UMTS (bottom)

In Table 2, we have reported the t_0 values when different wireless technologies are used to download files of three different sizes (5KB, 1MB, 2MB). Obviously, the values of t_0 are functions of $E(T)$ and μ_0 as reported in Table 2. Comparing our t_0 results with a t-student value equal to $t_{\alpha/2,n-1} = t_{0.01} = 2.46$, we can conclude that we have not rejected a valid model as the following inequality is satisfied:

$$|t_0| < t_{0.01}. \qquad (3)$$

Further, we have conducted a test hypothesis H_1 to compute the probability of accepting the model as valid when it is not valid:

$$\beta = P(H_0 \text{ failing to reject} \mid H_1 \text{ is true}). \qquad (4)$$

This test statistics has been conducted with the value of the difference between $E(T)$ and μ_0 computed by means of the following formula

$$\hat{\delta} = \frac{|E(T) - \mu_0|}{S}. \qquad (5)$$

In Table 3, we have reported the β values when different wireless technologies are used to download files of three different sizes (5KB, 1MB, 2MB). Obviously the values of β are function of (5). It is possible to observe that in most situations the probability of accepting a wrong model is satisfiable (10%). In a few situations this probability increases up to 25%. This occurs typically when the Wi-Fi technology is exploited. This result does not take us unaware as with this technology we have large variations in throughput due to different loads.

Table 2. Type I error ($\alpha = 0.01$)

Test n=30	5KB		1MB		2MB	
	$\hat{\delta}$	β	$\hat{\delta}$	β	$\hat{\delta}$	β
Wi-Fi	0.47	0.2	0.47	0.2	0.37	0.25
GPRS	1.70	0.1	1.36	0.1	1.21	0.1
UMTS	0.95	0.1	0.32	0.1	2.34	0.1

Table 3. Type I error (β)

Test n=30	5KB			1MB			2MB		
	μ_0	E(t)	T_0	μ_0	E(t)	t_0	μ_0	E(t)	t_0
Wi-Fi	0.20	0.22	0.02	18.1	18.5	0.26	32.4	32.9	0.20
GPRS	1.28	0.40	0.93	256.1	253.3	0.75	550.2	552.0	0.66
UMTS	0.60	0.20	0.52	101.0	101.6	0.17	211.8	205.2	1.28

CONCLUDING REMARKS

We have designed and implemented a coarse-grained network simulator (called Interceptor) which incorporates an emulation interface specifically developed to integrate into the simulation environment traffic coming from real network devices. An evident advantage of a coarse-grained approach, on which our simulator is based, stems from the fact that a large amount of the incoming traffic may be processed, through the emulative interface, without remarkable performance degradation in the simulation process. Based on three different wireless modeling exercises, we have presented evidence that Interceptor is an emulation-based platform suitable for modeling and simulating complex networked scenarios within a lab-based setting. In particular, a validation analysis has been carried out based on the type I and II errors theory that confirms the validity of our wireless modeling activity.

REFERENCES

J. Banks, J. S. Carson II, B. L. Nelson, D. M. Nicol, "Discrete-Event System Simulation", Pretience Hall, Upple Saddle River NJ, 2001

P. Barford, M.E. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation", Proc. of ACM Performance'98/SIGMETRICS, Madison, WI, 1998, 151-160.

C. Bettstetter, H.J. Vögel, and J. Eberspächer "GSM Phase 2+ General Packet Radio Service GPRS: Architecture, Protocols, and Air Interface", IEEE Communications Surveys, Third Quarter 1999, vol. 2 no. 3

L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K, Varadhan, Y. Xu, H. Yu, "Advances in Network Simulation", IEEE Computer, May 2000, 59-67.

V. Cardellini, E. Casalicchio, M. Colajanni, "A Performance Study of Distribuited Architectures for the Quality of Web Services", in proceeding of 34th Hawaii International Conference on System Sciences, 2001

K. Fall, "Network Emulator in the Vins/NS Simulator", Proc. of ISCC99, July 1999.

C. L. Ferris, F. E. Grubbs, and C. L. Weaver, "Operating Characteristics for the Common Statistical Test of Significance", Annals of Mathematical Statistics, 1946

R. Kalden, I. Meirick, M. Meyer, " Wireless Internet Access Based on GPRS" , IEEE Personal Communications, Vol. 7 N.2, April 2000, 8–18

M. Roccetti, V. Ghini & S. Cacciaguerra, "Interceptor: a Tool for Enabling Large-Scale Simulation of Software Distributed Applications in a Networked Laboratory", Proc. of 6th World Multiconference on Systemics, Cybernetics and Informatics (ISAS/SCI 2002), IIIS Press, Orlando (USA), July 2002, 242-246.

M. Roccetti, P. Salomoni, V. Ghini, S. Ferretti & S. Cacciaguerra, "Delivering Music over the Wireless Internet: from Song Distribution to Interactive Karaoke on UMTS Devices", chapter accepted for inclusion In Handbook of Wireless Internet, (B. Furht, M. Ilyas eds.), CRC Press, Boca Raton (USA), March 2003, 537-565.

P. Salomoni, "A Multimedia Mobile City Guide for Outdoor Learning", Proc. of 2003 ICSEE/Western MultiConference on Computer Simulation (ICSEE/WMC'03), Orlando (Florida), January 2003.

AN XML BASED NETWORK SIMULATION DESCRIPTION LANGUAGE

Roberto Canonico
Donato Emma
Giorgio Ventre
Università di Napoli Federico II
Dipartimento di Informatica e Sistemistica
Via Claudio, 21
I-80125 Napoli, ITALY
E-mail: roberto.canonico@unina.it

KEYWORDS
Web-based Network Simulation, XML description language

ABSTRACT

This paper presents an XML-based description language for describing generic network scenarios. We also present an XSL translation process that can be used to automatically translate a simulation scenario into a simulation script for a well known network simulator. This work is part of a larger project, aimed to define a framework, driven by this XML description language, that use more network evaluation technique for the study of a large network system.

INTRODUCTION

In recent years, simulation has become an indispensable technique to evaluate the performance of communication protocols and application for computer networks. Quite a large number of network simulators have been developed over the past decade.
However, the start of a simulation process, and the evaluation of its results, may require the use of information obtainable through the application of other evaluation approaches, such as direct measurement or testbed experimentation. In such a context, the availability of a single language to describe the architecture of a given network might be of great help. Such a description language should satisfy the following requirements:

- it should be independent from the analysis tool of choice;
- translation to/from the specific language adopted by the analysis tools should be easy and automatic;
- it should be adaptable to different scenarios (e.g. mobile hosts, wireless networks, and so on);
- it should be extensible.

These requirements have guided our work. We have defined an *XML based Network Simulation Description Language* to describe network scenarios that can be automatically translated into:
1. simulation scripts for a variety of network simulators;
2. analytical model;
3. testbed configuration scripts.

In this paper we first describe the XML description language that we have defined for describing newwork simulation scenarios. Later on, we present how we managed to automatically translate an XML simulation scenario into an *ns-2* simulation script. Next we present an example of use of our XML based language. Finally, we present some concluding remarks and future work.

DESCRIBING NETWORK SCENARIOS IN XML

A network scenario is a description of all the network components that need to be modeled when the behavior of a given network needs to be simulated. Basically, a network scenario is composed of:

- a description of the network topology;
- a description of the traffic flows that will be generated during the simulated time;
- a description of particular events (e.g. link failures) that need to be modeled in the course of the evaluation;
- a description of the expected output.

In this section we describe the basic concepts that make a network simulation scenario. To better illustrate the logical relations among the various components of a simulation scenario, we make use of UML class diagrams. In a second step, we illustrate how a simulation scenario can be described by means of an XML document. To describe the structure of such an XML document, we make use of XML Schema. In the process of translating the UML description into an XML document, we had to face the lack of typical O-O paradigms in XML.

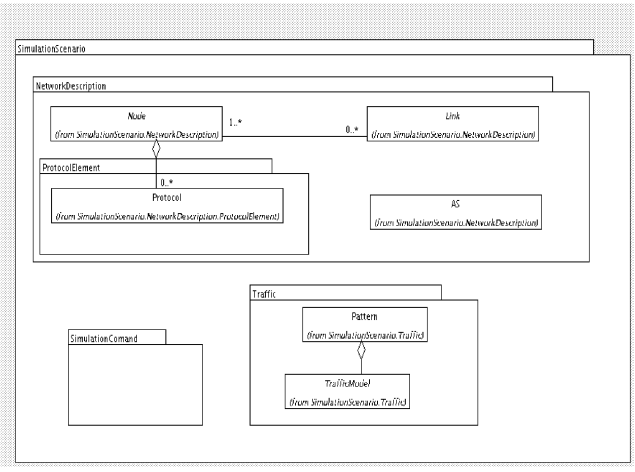


Figure 1: General Structure of a Network Scenario

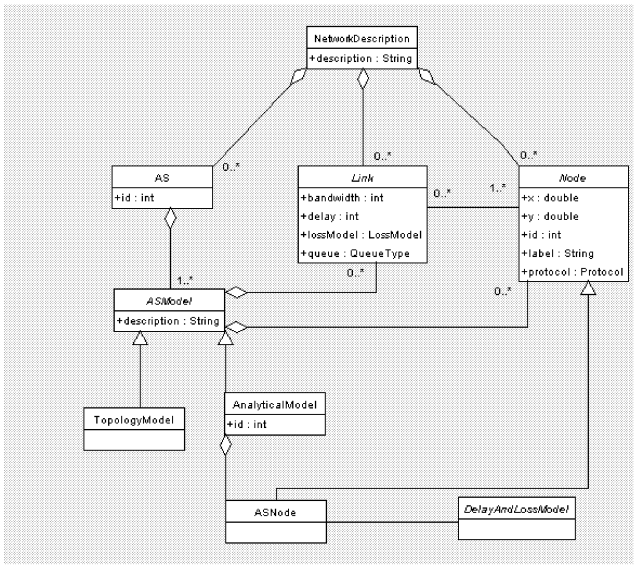


Figure 2: UML Class Diagram for the Network Description component

Figure 1 illustrates the main components of a Simulation Scenario in terms of abstract classes. Basically, it shows that a Simulation Scenario is made of three parts:
· a Network Description
· a Traffic Description
· a set of Simulation Commands

The Network Description is a static representation of the network topology. Hence, it is a collection of Nodes and Links. Here, the term "node" stands for "router or end-system". In a node, one or more instances of Network Protocol may be present.
Since modeling large scale networks is usually based on network partitioning at a domain level, the most important network abstraction is the "Autonomous System". Hence, at a topological level, a simulated network should be represented as a collection of inter-connected Autonomous Systems. However, we also take into account the possibility of simulating small networks in which the notion of AS is not present. For these smaller networks, the network description is simply made of network nodes connected by links.
Figure 2 shows the structure of a Network Description in greater detail. This picture shows that a Network Description is made of ASs, each of which is identified by its own ID. Autonomous Systems are interconnected by Links.
The internal structure of an AS is described by its AS model(s). The simplest description is by means of a "Topological Model", in terms of Nodes and Links. However, an AS can also be described by other formalisms, that we call "Analytical Models". All these formalisms have in common an "AS Node", embedding the formal representation of the AS behavior. Figure 2 shows that the concept of AS Node is derived from a generic "Node" class, which can also be used to model a generic network router and/or an end-system.
The Traffic Description component contains the information required to describe the traffic flows that will be generated during the simulation. Each flow may be characterized by a proper Traffic Model, which can be:

- an analytical model, describing the stochastic properties of the traffic flow;
- an application model, describing the specific traffic patterns generated by well-known applications (e.g. Telnet, HTTP, FTP, …);
- a traffic trace.

The latter part, the Simulation Command component, contains time-dependent events, such as variations in the network topology, in the routing policies, and so on.

XML SCHEMAS FOR THE PROPOSED DESCRIPTION LANGUAGE

In this subsection we illustrate how we have translated the class diagrams described in the previous section in an XML format, that we describe by means of XML Schemas (W3C Consortium 2001).
Figure 3 shows the format that we have defined for an XML document describing a network simulation scenario. Such a Schema is stored in a XSD file, which refers to three external XSD files: NetworkDescription.xsd, Traffic.xsd, and SimulationCommand.xsd. These three files contains the three components of a simulation scenario that we have described above. Fig. 4 shows the structure of the Network Description part of a simulation scenario. This document refers to external files (e.g. Node.xsd, Link.xsd, AS.xsd) that are not included in this paper, for the sake of brevity.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema attributeFormDefault="unqualified"
   elementFormDefault="qualified"
   targetNamespace="http://verde3.grid.unina.it/scenario"
   version="0.1"
   xmlns="http://verde3.grid.unina.it/scenario"
   xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:include schemaLocation="NetworkDescription.xsd"/>
<xs:include schemaLocation="Traffic.xsd"/>
<xs:include schemaLocation="SimulationCommand.xsd"></xs:include>
    <xs:element name="scenario">
        <xs:complexType>
           <xs:sequence>
            <xs:element name="networkDescription"
              type="NetworkDescription"/>
            <xs:element maxOccurs="1" minOccurs="0" name="traffic"
              type="Traffic"/>
            <xs:element maxOccurs="1" minOccurs="0"
              name="simulationCommand"
              type="SimulationCommand"/>
           </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Figure 3: Simulation Scenario XML Schema

Just to illustrate how this process of description need to be further refined, we present in Figure 5 the Node.xsd Schema, describing the format of the XML description of a simulated network node (e.g. a router or an end-system). As illustrated in Figure 2, the Node element may be specialized into a "ASNode" element, describing a whole network domain. This is useful for hybrid kinds of simulation, in which a simplified analytical model is used to describe the behavior of a complex domain (such as an Autonomous System) (Baumgartner et al. 2003).

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema attributeFormDefault="unqualified"
    elementFormDefault="qualified"
    targetNamespace="http://verde3.grid.unina.it/scenario"
    xmlns="http://verde3.grid.unina.it/scenario"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:include schemaLocation="Node.xsd"/>
<xs:include schemaLocation="Link.xsd"/>
<xs:include schemaLocation="AS.xsd"/>
    <!-- -->
    <!-- NetworkDescription Class Definition -->
    <!-- @param description -
            @param as - AS of the network
            @param node - Top level node
            @param link - Top level link
     -->
    <xs:complexType name="NetworkDescription">
      <xs:sequence>
         <xs:element name="description" type="xs:string"/>
         <xs:element maxOccurs="unbounded" minOccurs="0" name="as"
             type="As"/>
         <xs:element maxOccurs="unbounded" minOccurs="0"
             name="node" type="Node"/>
         <xs:choice maxOccurs="unbounded" minOccurs="0">
         <xs:element maxOccurs="unbounded" minOccurs="0"
             name="link" type="PointToPoint"/>
         <xs:element maxOccurs="unbounded" minOccurs="0"
             name="multiAccess" type="MultiAccess"/>
         </xs:choice>
      </xs:sequence>
    </xs:complexType>
</xs:schema>
```

Figure 4: Network Description XML Schema

```
   <xs:include schemaLocation="DataType.xsd"/>
    <xs:include schemaLocation="Protocol.xsd"/>
    <xs:include schemaLocation="CloudModel.xsd"/>
    <!-- Node Class Definiton
            @param id - Node Id
            @param label - Node Label
            @param x - Node x axis
            @param y - Node y axis
            @param protocol - Node Protocol graph
            @param ASNode - Subclass ASNode
     -->
    <xs:complexType name="Node">
        <xs:sequence>
            <xs:element minOccurs="0" name="label">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:pattern value="([a-zA-Z](_)?)*"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element maxOccurs="1" minOccurs="0" name="protocol"
                type="ProtocolList"/>
        </xs:sequence>
        <xs:attribute default="Router" name="type" type="NodeType"
           use="optional"/>
        <xs:attribute name="id" type="xs:nonNegativeInteger"
           use="required"/>
        <xs:attribute name="x" type="xs:float" use="optional"/>
        <xs:attribute name="y" type="xs:float" use="optional"/>
    </xs:complexType>
    <!-- AsNode Class Definition
        @parent Class Node
        @param model - Analitical Model of AS
    -->
    <xs:complexType name="AsNode">
        <xs:complexContent>
            <xs:extension base="Node">
                <xs:sequence>
                    <xs:choice maxOccurs="unbounded" minOccurs="0">
                    <xs:element name="cloudModel" type="CloudModel"/>
                    </xs:choice>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

Figure 5: Node Description XML Schema

TRANSLATING AN XML SCENARIO IN NS-2 SIMULATION SCRIPTS

The XSD Schemas we have presented in the previous section may be used to describe a complete simulation scenario in a simulator-independent way. Besides the XML Schemas, we have also defined a set of precise translation rules that can be used to produce a simulation script for the well known *ns2* network simulator (Breslau et al. 2000), starting from an XML simulation scenario. These translation rules have been expressed by means of the XSLT transformation language (W3C Consortium 1999).
The XSLT transformation document has been organized in independent files, according to the structure of the Schemas presented in the previous section. The XSLT files contain ns2 specific code and translates the XML simulation scenario in a script that can be executed by a ns2 simulator.

CONCLUSIONS AND FUTURE WORK

In this paper we have presented an XML description language that can be used to describe network simulation/emulation scenarios in a simulator-neutral way. Such an approach can be useful to allow interoperation of different network simulators, analytical model or testbed based approach to the evaluation of the behavior of complex network scenarios. We have also presented how we translate by means of XSLT our XML-based description language in simulation scripts for the ns-2 network simulator. We are currently investigating the difficulty of translating simulation scenarios formatted according to our specification in simulation scripts suitable for different network simulators.

ACKNOWLEDGEMENTS

This work has been partially carried out under the financial support of the Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) in the framework of the FIRB project "Middleware for advanced services over large-scale, wired-wireless distributed systems" (WEB-MINDS), and of the project "Scalability and Quality of Service in Web Systems".

REFERENCES

Breslau L.; D. Estrin; K. Fall; S. Floyd; J. Heidemann; A. Helmy; P. Huang; S. McCanne; K. Varadhan; Y. Xu; and H. Yu. 2000. "Advances in Network Simulation". *IEEE Computer*, Vol. 33 No. 5 (May), 59-67.

Baumgartner F.; M. Scheidegger; and T. Braun. 2003. "Enhancing Discrete Event Network Simulators with Analytical Network Cloud Models". *Proceedings of First international workshop on Inter-domain performance and simulation, IPS 2003*, Salzburg 20-21 February, Austria.

W3C Consortium. 2001. "XML Schema W3C Recommendation, 2 May 2001". [Online]
Available: http://www.w3.org/TR/xmlschema-0/

W3C Consortium. 1999. "XSL Transformations (XSLT) Version 1.0W3C Recommendation 16, November 1999". [Online].
Available: http://www.w3.org/TR/xslt

EXTENDED NAM: AN NS2-COMPATIBLE NETWORK TOPOLOGY EDITOR FOR SIMULATION OF WEB CACHING SYSTEMS ON LARGE NETWORK TOPOLOGIES

Roberto Canonico
Donato Emma
Giorgio Ventre
Università di Napoli Federico II
Dipartimento di Informatica e Sistemistica
Via Claudio, 21
I-80125 Napoli, ITALY
E-mail: roberto.canonico@unina.it

KEYWORDS
Network simulation, Topology generators, Web caching

ABSTRACT

Designing new network protocols for the Internet relies on the use of network simulation tools for validation. When simulation is aimed at testing the scalability of new protocols, working with large-scale realistic topologies is of paramount importance. In this paper we present an extension of the NAM Editor, a graphical editor to be used in conjunction with the well-known ns2 network simulator. Our extended NAM Editor allows the creation of synthetic but realistic network simulation scenarios and provides support for the simulation of web caching systems.

INTRODUCTION

Due to the high complexity of the modern Internet, the recourse to simulation tools for the evaluation of the effectiveness of new protocols and the feasibility of new applications, is of uttermost importance (Breslau et al. 2000). In particular, when simulation is aimed at testing the scalability of new protocols and applications, working with large-scale realistic network topologies becomes an indispensable requirement (Calvert et al. 1997; Cowie et al. 1999; Floyd and Paxson 2001).
In the last few years different network simulation tools have been developed. One of the most popular is the ns2 Network Simulator (Breslau et al. 2000). Ns2 is an open-source event driven network simulator. It provides support for simulation of IP-based network. In particular, ns2 provides researchers with:

- multicast and unicast routing protocols;
- different transport protocols (TCP, UDP, RTP, etc..);
- most common applications (FTP, Telnet, HTTP).

The ns2 official distribution is complemented by NAM, a graphical tool for visualization of simulation results (Estrin et al. 2000). More recently, NAM has been enriched with minimal editing functionality. This new tool, called NAM Editor, allows the interactive creation of simulation scenarios. The usefulness of NAM Editor, however, is quite limited, since its design capabilities are suitable for network of limited size only.
In order to simulate networks with realistic topologies, it is a common practice to use ad-hoc topology generators like *gt-itm* (Calvert et al. 1997; GT-ITM 2003) and *INET* (Jin et al. 2000). The output of these topology generators is usually not compatible with the ns2 syntax. Hence, several tools have been developed to translate topology descriptions generated by topology generators in ns2 scripts that can be used in the definition of a simulation scenario. Unfortunately, scripts produced in this way are not compatible with the NAM Editor, hence networks created by common topology generators cannot be modified interactively. Such a limitation is sometimes annoying, in particular when the automatically generated topology needs to be further adapted, e.g. by instantiating agents on particular network nodes.
Once the network topology has been generated, the simulation scenario must be completed with traffic data. This requires the definition of proper traffic patterns (according to the characteristics of the application that is being simulated), and the localization of traffic end-points (sources and sinks). In a complex simulation scenario (e.g. composed by hundreds or thousands of nodes) localizing traffic end-points is not a trivial operation, which can be usefully supported by a graphical interactive tool like the NAM Editor.
To overcome the above limitations we have implemented an Extended version of the NAM Editor with the following features:

1. integration of existing topology generators;
2. localization and visualization of sets of nodes on large network topologies according to different selection criteria;
3. instantiation of *agents* (i.e. the elements that implement transport-level or application-level functionality in ns2 terminology) of any type on all the nodes of a given node set;
4. customization of node parameters;
5. definition of new node types (e.g. web caches);
6. support for simulation of web cache systems.

The main features of our tool will be described in the following sections.

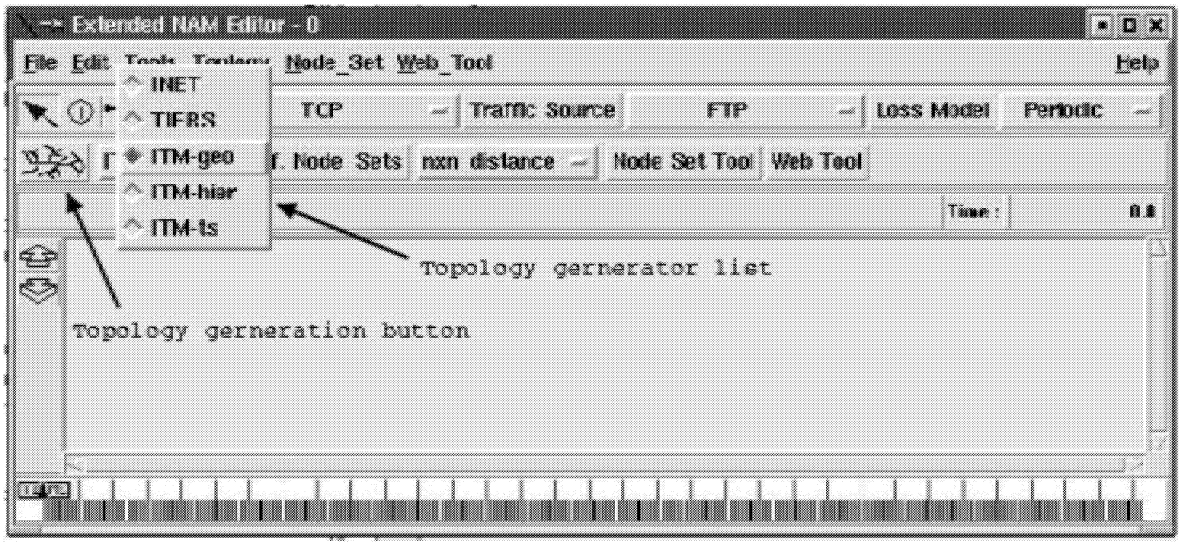


Figure 1: Topology Generation within Extended NAM Editor

INTEGRATION OF TOPOLOGY GENERATORS

Our Extended NAM Editor makes it possible to create, visualize and modify topologies created according to the following topology models:

- flat-itm;
- hierarchical-itm;
- transit-stub-itm;
- INET model.

Our tool not only provides the topology generators with the required input parameters, but it also translates the generated topology description in the internal representation of a network used by the NAM Editor.

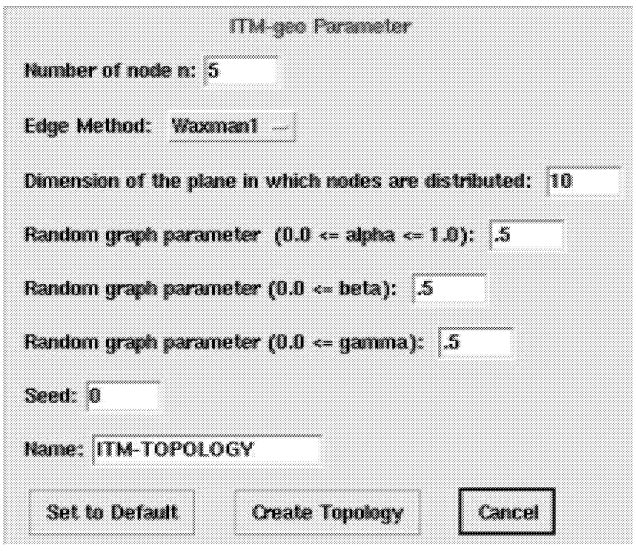


Figure 2: Input Parameters for the ITM-geo Topology

Figure 1 shows how this feature has been integrated in the NAM Editor GUI. Figure 2 shows the input parameters the user has to provide in order to generate a network topology with the gt-itm generator.

LOCALIZATION OF NODE SETS

Localization of a network element (*agent* in ns2 terminology) in a simulation scenario is a two steps process, which requires:

- the localization of the node in which the agent must be instantiated
- the instantiation of the agent.

To support agent instantiation in large topologies we have included in the extended NAM Editor the concept of *Node Set*. A Node Set is a set of nodes selected according to one of the following criteria:

- leaf node;
- mutual distance;
- randomly.

When a network topology is created with one of the supported topology generators, some Node Sets are automatically created, reflecting the topology model. For instance, in the case of a transit-stub topology, a Node Set is associated to each transit domain and a different one to each stub domain.
To make this tool even more flexible, we made the GUI of our Extended Nam Editor customizable by the end user. Thanks to this feature, it is possible to:

- add an user-defined agent to the agent list;
- customize the parameters of a user-defined agent;
- define new node type and its parameters

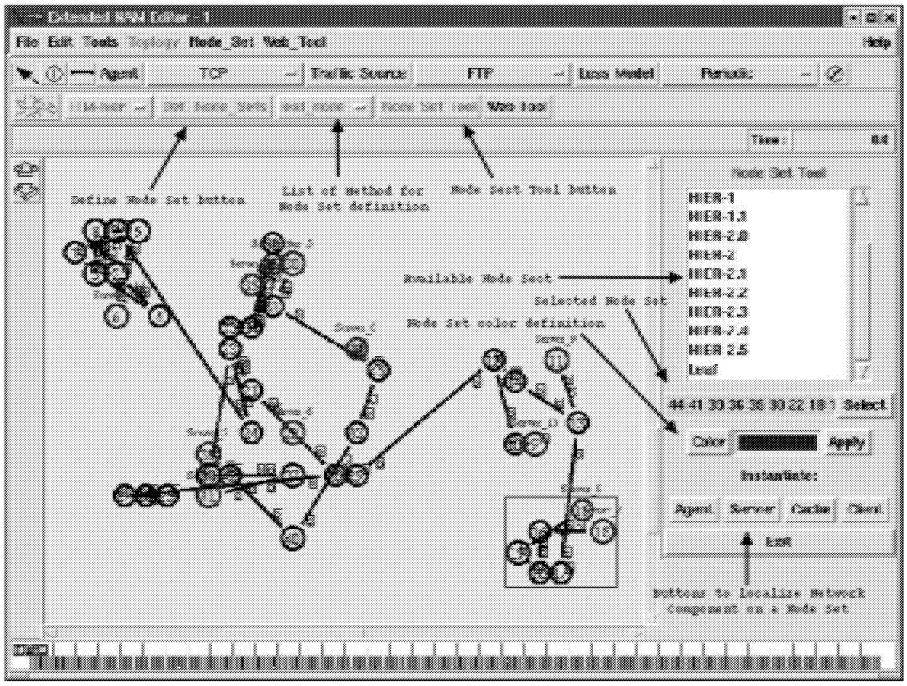


Figure 3: The Node Set Tool in the Extended NAM Editor

Features described in this section are implemented by the Node Set Tool (Figure 3) which allows to:
1.visualize all the nodes included in a given set;
2.associate a color to all the nodes included in a given set;
3.instantiate an agent on all the nodes of a given set;
4.instantiate a Web Server on all the nodes of a given set;
5.instantiate a Web Client on all the node of a given set;
6.instantiate a Web Cache on all the node of a given set.
The latter three functionalities, in particular, are meant to support the simulation of web cache systems on large network topologies.

SIMULATION OF WEB CACHE SYSTEMS

A special support as been provided in our Extended NAM Editor for simulating web cache systems. A proper tool can be used to create web-page pools and to deploy system of caches, servers and clients onto a previously generated topology. The configuration of any web element is made through a configuration window.
The combined use of this functionality with the possibility of generating realistic large scale topologies makes this tool extremely useful for the simulation of Content Delivery Networks or other systems in which management of replicated content is done through distributed caches.

CONCLUSIONS

In this paper we have presented an interactive graphical editor that can be used as a front-end for the well-known ns2 network simulator. Our tool integrates most popular network topology generators and also provides users with several features that are extremely useful for simulating large-scale systems of web caches and Content Delivery Networks.

ACKNOWLEDGEMENTS

This work has been partially carried out under the financial support of the Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) in the framework of the FIRB project "Middleware for advanced services over large-scale, wired-wireless distributed systems" (WEB-MINDS), and of the project "Scalability and Quality of Service in Web Systems".

REFERENCES

Breslau L.; D. Estrin; K. Fall; S. Floyd; J. Heidemann; A. Helmy; P. Huang; S. McCanne; K. Varadhan; Y. Xu; and H. Yu. 2000. "Advances in Network Simulation". *IEEE Computer*, Vol. 33 No. 5 (May), 59-67.

Calvert, K.; M. Doar; and E. W.Zegura. 1997. "Modeling Internet Topology". *IEEE Communications Magazine*, Vol. 35 No. 6 (June), 160-163.

Cowie J. H.; D.M. Nicol; and A.T. Ogielski. 1999. "Modeling the Global Internet". *Computing in Science & Engineering*, Vol. 1 No. 1 (Jan), 42-50.

D. Estrin; M. Handley; J. Heidemann; S. McCanne; Y. Xu; and H. Yu. 2000. "Network Visualization with Nam, the VINT Network Animator". *IEEE Computer*, Vol. 33 No.11 (Nov), 63-68.

Floyd, S. and V. Paxson. 2001. "Difficulties in Simulating the Internet". *IEEE/ACM Transactions on Networking*, Vol. 9, No. 4, (Aug), 392-403.

GT-ITM. 2003. Modeling Topology of Large Internetworks. University of Georgia Tech.
Available at: http://www.cc.gatech.edu/projects/gtitm/

Jin, C.; Q. Chen; and S. Jamin. "Inet Topology Generator". Technical Report CSE-TR-433-00, EECS Department, University of Michigan, 2000.

A GENERIC FRAMEWORK FOR PERFORMANCE TESTS OF DISTRIBUTED SYSTEMS

Frederic Hancke, Tom Dhaene, Frans Arickx and Jan Broeckhove
Department of Mathematics and Computer Science
University of Antwerp, Belgium
{frederic.hancke, tom.dhaene, frans.arickx, jan.broeckhove}@ua.ac.be

KEYWORDS
Distributed Computing, Farmer-Worker, JavaSpaces.

ABSTRACT

More and more different platforms allowing distributed computing make their appearance in literature. This induces the need for standardised testing techniques. This contribution describes the design and implementation of a framework based on the farmer-worker pattern using tasks and results for communication, uncoupled from the underlying distributed system. The presented framework firstly allows easy building of farmer-worker applications and secondly easy setup of testcases.

INTRODUCTION

In our research group CoMP (*Computational Modelling and Programming*) research is done in the field of computational science, aimed at understanding physics and engineering problems through modern modelling techniques, using new software development paradigms and advanced mathematical techniques.

Many problems, as in the area of quantum physics, often involve very intense and large computations. Therefore, distributed platforms, such as JavaSpaces [FHA99], MPICH [MPI] (an MPI [GES99] implementation), etc. provide a better and faster solution. The goal of our project is to characterise different existing platforms for certain kinds of calculations.

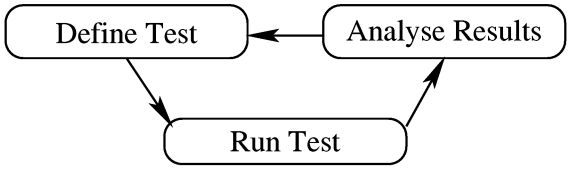


Figure 1: Cycle for testing distributed systems

Figure 1 illustrates the cycle for testing distributed systems. The first step is to define the test to be performed, the second to run the test and the third to analyse the results obtained from the test. The analysis may detect interesting areas for further investigation, hence the arrow concluding the cycle.

This contribution mainly focusses on the second step of this cycle. In order to run tests, the design and implementation of a framework will be discussed. This framework provides a good base to build applications and testcases based on the farmer-worker topology with use of the Linda tuple spaces paradigm [Yal]. Finally, an example coupling JavaSpaces to this framework is discussed.

FRAMEWORK

In our research activities we are often confronted with the question which distributed platform to choose for a certain calculation, often based on its brute performance. Therefore, different distributed systems have to be set up and tested. In the past, separate implementations for different platforms had to be made. The framework presented in this section has been developed to minimize the extra work when writing a test for a different, previously not tested, distributed platform.

The framework being introduced here has been developed in Java using the farmer-worker pattern with a communication concept of tasks and results. The first being distributed by the farmer and processed by the worker. The latter being returned by the worker upon finishing task processing, and collected by the farmer. The underlying communication system is based on the tuple spaces paradigm. Thus, tasks and results are being *sent* between farmer and workers using the concept of a space with basic operations read, write and take. This means farmer and worker do not communicate directly with each other, but rather via a space, although this could be forced using appropriate task descriptions. As distributed systems other than tuple space implementations could also be plugged in, at least when providing the three basic operations, the framework will not be perfect for these. The reason for this is of course that distributed platforms such as MPICH use concepts (like direct message passing) that are very different from tuple spaces.

Architecture

The architecture of the framework has been built in layers (see figure 2). As previously mentioned, the framework is implemented in Java. The first layer around Java provides the structure for building farmer-worker applications using a task-result communication via a space. The outer layer provides a frame for testing platforms using XML based [McL01] sleep tasklists [HSD+03]. Both layers are discussed in more detail in the rest of this section.

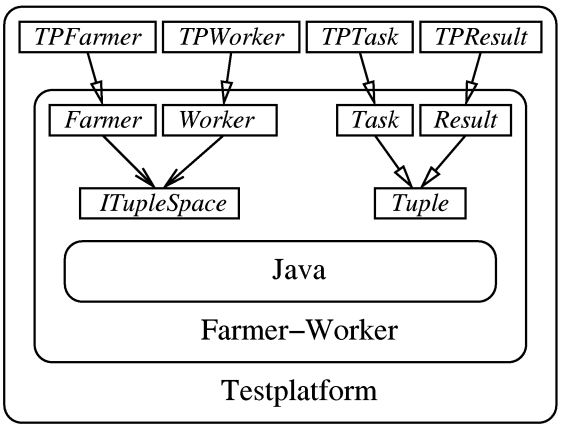


Figure 2: Framework architecture and partial class diagram

Farmer-Worker Layer

The Farmer-Worker layer provides a set of abstract classes to allow an easy way of building distributed applications based on the farmer-worker pattern using the concept of tasks and results, completely platform independant.

The abstract classes `Task` and `Result` (extended from `Tuple`) are available for extension in applications using this layer. Task and result descriptions can thus be defined depending on the applications needs. These abstract tuples are used by the abstract classes `Farmer` and `Worker`. Both communicate through the `ITupleSpace` interface — declaring the basic operations `read()`, `write()` and `take()` — and provide a method `setup()`, that is used for a priori setup (e.g. connections to be established, variables to be set, etc. before the real process is allowed to start), and a method `run()`. The `run()` method in the worker starts the task processing cycle (which is looped infinitely):

1. Wait for a `Task` object being taken from the space.
2. Process the task and get the result through the abstract method `handleTask()`.
3. Write the obtained `Result` object back to the space.

The farmer starts a slightly different process through the method `run()`. Two threads are started. One for distributing tasks, the other for collecting results. The method `run()` of the thread `DistributeTasks` performs the following steps (as many times as there are tasks):

1. Get a task through the abstract method `getNextTask()`.
2. Write the obtained `Task` object to the space.

The method `run()` of the thread `CollectResults` performs the steps (also as many times as there are tasks):

1. Wait for a `Result` object being taken from the space.
2. Process the result through the abstract method `handleResult()`.

Testplatform Layer

The Testplatform layer also provides a set of abstract classes, built around the Farmer-Worker layer, but specifically implemented for easy testing of distributed platforms. The available classes cover the use of the XML based tasklists (of sleeping times) and specific tasks and results, still platform independant.

The abstract classes `TPTask` and `TPResult` (extended from `Task` and `Result` respectively) each contain a data member for the duration to sleep. The abstract classes `TPFarmer` and `TPWorker` implement their respective inherited abstract method `handleResult()`, which basically does nothing, and `handleTask()`, which performs the actual sleep. In addition, the farmer performs the conversion from the XML based tasklist description to a linked list representation. As the application using the Testplatform layer typically needs a set of tasklists (XML files) to be run more than once for the total test, the farmer in this layer uses a utility to randomize the execution sequence of the tasklists (for example based on a randomized block design with replication as block criterium [NKNW96]).

Logging

Both layers discussed above have been implemented with the use of the logmanager introduced in Sun Microsystems' Java Platform 1.4 [HC03, HC02]. The main purpose of this logmanager is to eliminate the traditional `System.out.println()` typically used to debug applications. The `LogManager` basically holds a set of `Loggers` that contain methods to log certain messages (e.g. upon entering or exiting methods, instantiation of objects, variable information in large algorithms, etc.). These log messages can be assigned log levels. There are seven built-in log levels:

- SEVERE for indicating serious failures.
- WARNING for indicating potential problems.
- INFO for informational messages.
- CONFIG for static configuration messages.
- FINE for providing tracing information.

- FINER for indicating fairly detailed tracing information.
- FINEST for indicating highly detailed tracing information.

Via the use of handlers these log messages can be directed to the console, an XML file, etc., using a personal specified format through a formatter if desirable. The level of log messages to be effectively logged can be set in a small configuration file. The methods for logging messages will always be invoked, but will immediately return if the messages log level is not high enough to be logged according to the configuration file.

EXAMPLE

As an example to use the framework discussed in the previous section, an application for testing JavaSpaces (a Linda implementation based on Jini [Edw01]) was built around the Testplatform layer (see figure 3).

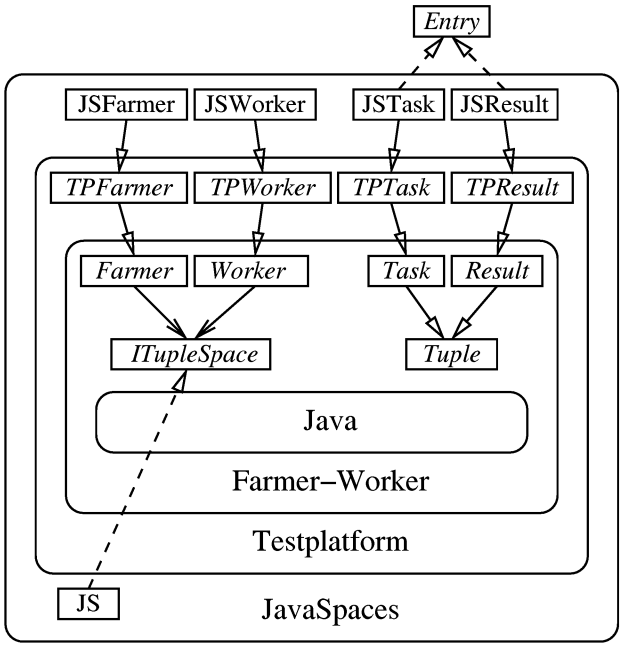


Figure 3: JavaSpaces example class diagram

First the `ITupleSpace` interface methods `read()`, `write()` and `take()` must be implemented (`JS`). Then the tuples `TPTask` and `TPResult` must be extended to tuples JavaSpaces can use. Therefore, both `JSTask` and `JSResult` must also implement the `Entry` interface provided by Sun Microsystems' JavaSpaces package. And finally the `TPFarmer` and `TPWorker` must be extended to implement some abstract methods, including `getTupleSpace()` which returns an instantiation of `JS`.

In the same way other distributed platforms can easily be coupled to the presented framework. For applications only needing the Farmer-Worker layer the implementation happens exactly the same way, except for a couple of abstract methods, which should not cause a problem.

CONCLUSIONS

This contribution discussed the design and implementation of a framework created to minimise efforts for building distributed applications using the farmer-worker pattern with tasks and results. An extra layer was added to extend the basic framework for easily testing distributed systems. The framework described is written in Java and completely uncoupled from the underlying distributed platform. As an example, JavaSpaces was coupled to the framework. It is clear that the framework presented here will be of great use when farmer-worker applications or testcases must rapidly be built or ported to other distributed platforms.

REFERENCES

[Edw01] W. Keith Edwards. *Core Jini*. Prentice Hall, second edition, 2001.

[FHA99] E. Freeman, S. Hupfer, and K. Arnold. *JavaSpaces Principles, Patterns, and Practice*. Addison Wesley, 1999.

[GES99] William Gropp, Lusk Ewing, and Anthony Skjellum. *Using MPI*. The MIT Press, second edition, 1999.

[HC02] Cay Horstmann and Gary Cornell. *Core Java 2, Volume II: Advanced Features*. Prentice Hall PTR, fifth edition, 2002.

[HC03] Cay Horstmann and Gary Cornell. *Core Java 2, Volume I: Fundamentals*. Prentice Hall PTR, sixth edition, 2003.

[HSD+03] Frederic Hancke, Gunther Stuer, David Dewolfs, Jan Broeckhove, Frans Arickx, and Tom Dhaene. Modelling overhead in javaspaces. In *Proceedings Euromedia*, pages 77–81. Eurosis, 2003.

[McL01] Brett McLaughlin. *Java & XML*. O'Reilly & Associates, Inc., 2001.

[MPI] Mpich. URL: http://www-unix.mcs.anl.gov/mpi/ mpich/.

[NKNW96] John Neter, Michael H. Kutner, Christopher J. Nachtsheim, and William Wasserman. *Applied Linear Statistical Models*. WCB/McGraw-Hill, fourth edition, 1996.

[Yal] Yale linda group. URL: http://www.cs.yale.edu/ Linda/ linda.html.

AI
AND
SIMULATION

AI AGENTS

A DYNAMIC ENVIRONMENT SIMULATOR

Xiangdong An
School of Computer Science
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1
Email: xan@cs.uwaterloo.ca

Yang Xiang
Dept. of Computing and Information Science
University of Guelph
Guelph, Ontario
Canada N1G 2W1
Email: yxiang@cis.uoguelph.ca

KEYWORDS
Simulation Software Tool, Graphical Model Editor, Dynamic Environment, Probabilistic Reasoning, Dynamic Multiagent Reasoning.

ABSTRACT

Simulated dynamic environments are needed in experimental study of algorithms for probabilistic reasoning in dynamic domains. Since it can be assumed that all professionals in information technology and many in science and engineering have a basic understanding of digital electronics, sequential digital circuits are simulated in this paper to satisfy such demands. Existing digital circuit simulation software displays circuit signals in a logic analyzer style window, which does not allow the reasoning agents to observe. In particular, these simulators are implemented for designing, simulating, and analyzing analog, digital and mixed electronic circuits, which do not allow faulty devices in the simulated circuits. Some of them are very expensive. We implement a simple sequential digital circuit simulator which is specifically used for probabilistic reasoning in dynamic domains. Our simulator can synchronously interact with more than one agents, hence, it can also be used as a test bed for dynamic multiagent reasoning.

1 INTRODUCTION

The world keeps changing. We have to study dynamic environments. However, experiments on real environments are very expensive at most times. For example, dynamic Bayesian networks (DBNs) (Dean and Kanazawa, 1988; Kjarulff, 1992; Russell and Norvig, 1995; Binder et al., 1997; Kuenzer et al., 2001; Murphy, 2002) are used for uncertain reasoning in dynamic domains, which can be applied in such real domains as water purification plant (Jensen et al., 1989) and freeway traffic (Forbes et al., 1995). We may not have permission to experiment on these domains. Even we have permission, it is costly to install sensors and collect observations about the environments. We may also need significant background domain knowledge on these domains to understand them.

Experiments would become convenient and cheaper if dynamic domains, which do not need significant technical knowledge to understand, can be simulated by computers. Since it can be assumed that all professionals in information technology and many in science and engineering have a basic understanding of digital electronics, sequential digital circuits are proposed to be simulated in this paper. A digital circuit can be combinatorial or sequential. In a combinatorial digital circuit, output values depend only on the input values, whereas in a sequential digital circuit, output values also depend on the previously stored values. Therefore, a sequential digital circuit provides a dynamic domain. Existing digital circuit simulation software is developed for designing, simulating, and analyzing analog, digital and mixed electronic circuits, which does not allow the reasoning agents to observe circuit signals (e.g. MacroSim, TINA, etc). They do not allow faulty devices in the simulated circuits. Some of them are very expensive (e.g. TINA). Software working on VHDL program (e.g. Maxplus) does not allow faulty devices in the simulated circuits and have difficulty to observe circuit signals.

Simulation of dynamic domains has been used in experimental study in probabilistic reasoning area (Jensen et al., 1989; Forbes et al., 1995; Boyen and Koller, 1998). The paper (Boyen and Koller, 1998) uses the simulated examples in the papers (Jensen et al., 1989; Forbes et al., 1995) where observations are simulated by sampling the evidence variables according to the exact distributions. The simulated examples in the two papers (Jensen et al., 1989; Forbes et al., 1995) have some fixed structures. In the case of paper (Forbes et al., 1995), users have to specify a variety of behaviors simulating good drivers, antisocial, incompetent, unsafe drivers and stalled vehicles, and so on. Users also need to specify when and how often they undertake such behaviors, their speed and so on. The simulator uses vehicles' decision at each "clock tick" to plot trajectories and to simulate collisions and other significant events. These make simulation complicated. In the case of paper (Jensen et al., 1989), users need to have technical knowledge in the domain of water purification.

Our simulator can design and simulate the running of sequential digital circuits where some of devices (e.g. gates, flip-flops) may be faulty. The simulator displays signal changes at all circuit signal points. Any changes in

the simulated dynamic domains can be directly viewed by users. The simulator can respond "observation" requests from more than one agent synchronously, which makes it available for experimental study in dynamic multiagent reasoning (Xiang, 2002; An, 2003). Our simulator contains two parts: the sequential digital circuit Editor and the sequential digital circuit Evaluator. We can draw a sequential digital circuit using the Editor and then simulate its running using the Evaluator.

2 DESIGN

2.1 Requirements

In order for agents to monitor a dynamic environment, they need to reason about events as well as the time when the events occur. In particular, when an agent observes an event, in general it needs to note the time of event. Assume that the environment is associated with a centralized clock. When an agent observes an event, it can observe the time of the event from this clock as well.

When we replace the environment by a simulator, we need to achieve the same effect, namely, allowing agents to associate events with the correct time. If the simulator can have large enough storage, this can be done by the simulator to record down the sequence of all events and their time. The agents can then read the events from this record. However, this method is impractical for unbounded period of time.

The alternative is for the simulator to generate events on the fly and allow the agents to "observe" the events as they "occur" and before they "disappear". This requires that agents' computation progresses with the same steps with the simulator. In particular, agents' computation should proceed according to the time maintained by the simulator's clock. When this is the case, we say that the agents are synchronized with the simulator.

2.2 Protocol

The following message passing protocol realizes the requirements.

(1) In one clock cycle, there exist only two message passing communications between the simulator and each agent. One is from the agent to the simulator and the other from the simulator to the agent. The message from an agent to the simulator contains all the agent's observation requests. The message from the simulator to the agent replies the agent's requests.

(2) Simulator won't go to next clock cycle before all agents on the current cycle are responded.

(3) Agents, which have got responses for the current clock cycle, may submit their new requests for the next clock cycle. However, new requests won't be responded until simulator replies all requests on current cycle and proceeds to next clock cycle.

(4) Agents move to the next cycle only when their requests are responded.

(5) Since simulator needs to know which coming requests correspond to the current time, simulator must maintain a clock and the message from agents to the simulator must contain timing information.

We illustrate the protocol with an example. Without losing generality, we have one simulator and 3 reasoning agents in our example. We use S to represent the simulator, A_0 agent 0, A_1 agent 1 and A_2 agent 2.

Initially, S stays there to wait for any requests.

In the first cycle of the clock, A_0, A_1 and A_2 send their requests to S individually. S accepts and processes all these requests. After responses are formed for an agent, S sends them to the corresponding agent. An agent sends its next set of requests to S after receiving the responses to its current set of requests. All agents may not get their responses to the requests on the same cycle of clock simultaneously. Agents, which get early responses, may send S their requests corresponding to next cycle of clock early. For example, A_0 and A_2 may get the responses to their requests before A_1 does. Therefore, A_0 and A_2 may send their new requests to S before A_1 does. However, the new requests cannot be processed before A_1 get the responses to its requests, because S is still in the old state (on the old cycle of clock). So, starting from the second cycle, we need a data structure to store all new requests which wait to be processed. After A_1 is responded, S goes forward to the next clock cycle. Then S processes all stored (from A_0 and A_2) and coming requests (from A_1) on the next cycle. In the second cycle, A_0 and A_2 may be served before A_1 sends its request. In this situation, the requests sent by A_0 and A_2 for the third cycle may come with the request by A_1 for the second cycle. S knows which coming requests correspond to the current cycle by comparing the current time it maintains and the timing information in the message from agents. Requests for the next cycle will be stored and request for the current cycle processed.

The process will be repeated cycle by cycle.

2.3 Key Properties

From the example above, we have the following conclusions:

- The agents are synchronized with the simulator by message passing.
- Between the simulator and each agent, two communications are used to synchronize all agents and the simulator, which is minimum for information exchanging between two agents.
- The number of agents waiting to be processed is at most $n - 1$ at each cycle of clock (n agents in total).

The subsequent sections will show our simulator also has the following properties:

- Faulty devices in the simulator can be specified by users which are visualized by elliptical nodes in the designed circuits.

- If a device is faulty, it may give arbitrary outputs which could be correct or incorrect. The percentage that the output of a faulty device could be correct can be specified by users. The output of a faulty device may also stick on some level which can also be specified by users.

- Signal level corresponding to each circuit signal point at each cycle of clock is visualized on the circuit.

- Observability of a variable can be set by users and is visualized.

3 IMPLEMENTATION

3.1 Elementary Devices

We implement the simulator by JAVA. To ensure the simulator is both simple and good enough to use, we make some simplification.

A sequential digital circuit is composed of basic logic gates (*and*, *or*), inverting logic gates (*inverter*, *nand*, *nor*), exclusive logic gates (*xand*, *xor*, *xnor*) and flip-flops (*S-R* flip-flop, *D* flip-flop, *J-K* flip-flop). Since *nand*, *nor* and *xnor* can be inverted from *and*, *or* and *xor* respectively by adding an *inverter*, and *S-R* flip-flop can be replaced by *J-K* flip-flop, we treat *and*, *or*, *xor*, *inverter*, *D* flip-flop and *J-K* flip-flop as elementary devices in the simulated circuits.

3.2 Representation on Display

Green nodes are used to represent circuit signal points. Green nodes with a letter inside are used to represent devices: a green node with a letter '*A*' inside represents an *and* gate; a green node with a letter '*O*' inside represents an *or* gate; a green node with a letter '*I*' inside represents an *inverter*; a green node with a letter '*X*' inside represents an *xor* gate; a green node with a letter '*D*' inside represents a *D* flip-flop; a green node with a letter '*J*' inside represents a *J-K* flip-flop. Devices are connected by links. Arrows on the links represent the direction by which signals flow. A green bar beside a circuit signal point denotes its current level is *1*; otherwise *0*. Since *J-K* flip-flops have two inputs with different meaning (*J* and *K*), we propose to represent link *J* with a special color (e.g. cyan). Flip-flops have two opposite outputs: Q and $\bar{Q}$. A special color (e.g. magenta) indicates output Q. Normal link color is black.

A signal node with an other than green color (e.g. pink) is considered unobservable. The status of a device can be represented by the shape of the node. A green ellipse stands for an abnormal device and a green circle a normal device. By default, all nodes drawn are signal nodes which can be converted into device nodes by clicking corresponding buttons.

3.3 Data Structure

We have a class *DCNode* (Digital Circuit Node) to represent nodes in a digital circuit. A digital circuit is an array of instances of *DCNode*. Below is part of the class *DCNode* that shows all variables used in the class.

```
Class DCNode {
    type // device type: and, or, J-K flip-flop ...
    status // normal or faulty
    label // name of the variable
    position // position on window
    value // signal level
    preStatus // previous state for flip-flops
    observability // if observable
    parent // parent list
    child // child list
    JIndex // parent coming as input J of a J-K flip-flop
    QIndex // output Q (versus Q̄) of a flip-flop
}
```

3.4 Synchronization and Deadlock

Since asynchronous digital circuits (e.g. ripple counters) has the problem of the accumulated propagation delay of the clock from flip-flop to flip-flop, we simulate synchronous digital circuits in this paper. That is, all flip-flops are controlled by the same clock signal. Hence, the clock signal is ignored in the simulator. The simulator and all inquiring agents are synchronized by message passing. All inquiring agents communicate with the simulator through TCP/IP protocol. Multiple threads are used to respond all inquiring agents. One thread is responsible to one inquiring agent at one cycle of clock. After responding to an inquiry, the thread is terminated.

If an early inquiry comes, a thread is started but won't answer until the simulator proceeds to next cycle. By this way, we stored early inquires. Before a thread is started, we need to check if all inquires on current cycle have been responded. If all inquires on current cycle have been responded, the simulator will go to next cycle. If the responses to the last inquiring agent are sent before marking that it is responded, deadlock may happen: when checked, the agent is not marked indicating that it is responded, the simulator waits for the agent for current cycle, which won't send requests for current cycle. The simulator won't go to next cycle and the time won't be updated. All agents that are waiting processing won't proceed. It can be solved by always marking that the agent is served before sending responses to the agent.

4 EDITOR

4.1 An Example Drawn by the Editor

Figure 1 shows the interface of the Editor and a sequential digital circuit drawn by the Editor. The digital circuit con-

tains one *D* flip-flop, three *J-K* flip-flops, four *and* gates, one *or* gate and one *inverter*.

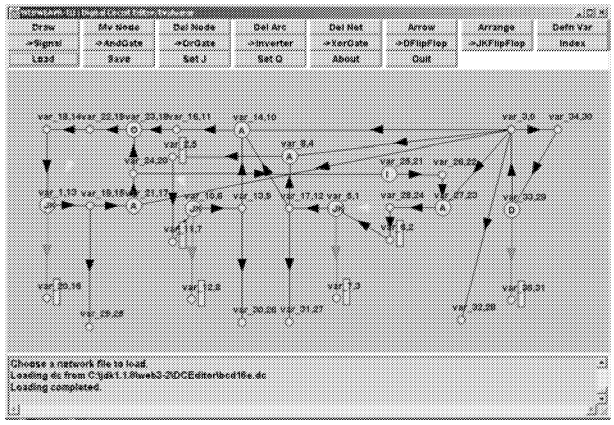


Figure 1: A sequential digital circuit drawn by the Editor.

4.2 Drawing

By clicking button "Draw", we can draw a network where nodes represent signals. To make it a digital circuit, we can convert a node into a device by clicking corresponding Editor buttons.

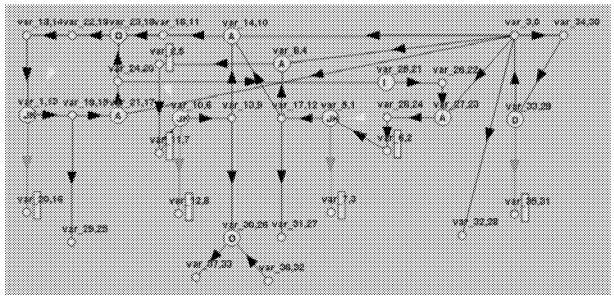


Figure 2: Signal node var_30 converts to an *or* gate.

For example, by first clicking button "→OrGate" and then clicking node *var_30* in the digital circuit in Figure 1, the signal node becomes an *or* gate in Figure 2. By default, the *or* gate have two inputs and one output. Suppose the *or* gate is used to "*or*" *var_13* and *var_17*. Click "Draw" button, a link between *var_30* and *var_17* is drawn. Click "Del Node" button and then click *var_36*, the redundant node is removed. Thus the *or* gate is properly inserted. Click "Del Node" button, and then click node *var_31*, the node *var_31* is also removed (see Figure 3). By clicking "Set J" button and then clicking the link between *var_5* and *var_6*, the two inputs of the *J-K* flip-flop *var_5* in Figure 2 switch as in Figure 3. The cyan color denotes the link is input '*J*' of a *J-K* flip-flop. By clicking "Set Q" button and then clicking the link between *var_17* and var_5, the two outputs in Figure 2 switch as in Figure 3. The magenta color denotes the link is output '*Q*' of a flip-flop. Button "Mv Node" can be clicked to adjust nodes' positions. "Del Arc" can be used to remove links.

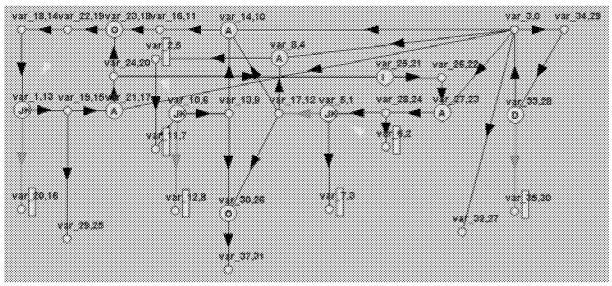


Figure 3: Inputs of *J-K* flip-flops and outputs of flip-flops can be set.

In Editor, user can specify variable labels, external input values and initial states of the flip-flops. User can also specify which signal points and devices can be observed and which devices are faulty. These specifications are finished through button "Def Var". If a device is faulty, its output could be arbitrary. The output could be correct or incorrect. The percentage that a faulty device gives a correct output can be specified by user by clicking button "Def Var" and then clicking the device. Percentage *100* denotes the device is normal. A percentage p between *0* and *100* denotes the device is faulty and will give correct output by $p\%$. If a flip-flop is faulty, one or both two of its outputs (Q, $\bar{Q}$) may be abnormal. A faulty device may be stuck at a constant level. We can also specify the situation through button "Def Var". By clicking "Def Var" button and the device, instead of a number between *0* and *100*, we use *-1* to denote its output will be stuck at '*0*' and *-2* to denote its output will be stuck at '*1*'. If a device is faulty, the node will be represented by a corresponding green ellipse instead of a corresponding green circle.

5 EVALUATOR

Figure 4 shows an Evaluator which has loaded a sequential digital circuit where all external inputs and initial states of all flip-flops have been specified. All internal outputs are designated to be *0* by default. The real values of all internal outputs have to be figured out based on external inputs and states of flip-flops. Button "Agents" is used to specify the number of agents to communicate with the simulator. Once button "Run" is clicked, the digital circuit goes into its first state - state 0 where the values of all variables are updated based on preceding external inputs and states of flip-flops. For example, the output Qs (*var_20*, *var_35*) of two normal flip-flops *var_1* and *var_33* are specified to be *1* and their output $\bar{Q}$s (*var_19*, *var_3*) should be *0* consequently. Hence, the result *var_24* of *var_19* *and var_3* is *0* and its inverse *var_26* changes from *0* in Figure 3 before updating to *1* in Figure 5 after updating.

Before all inquiring agents communicate with the simulator, the simulator won't go into its next state - state 1. An agent will send its requests for next cycle of clock after it receives the responses to its current requests. The simulator only replies requests when it proceeds into the same

cycle of clock with corresponding agents. After all agents are responded for current cycle of clock, the simulator proceeds into next cycle of clock and all variable values are updated based on the preceding state.

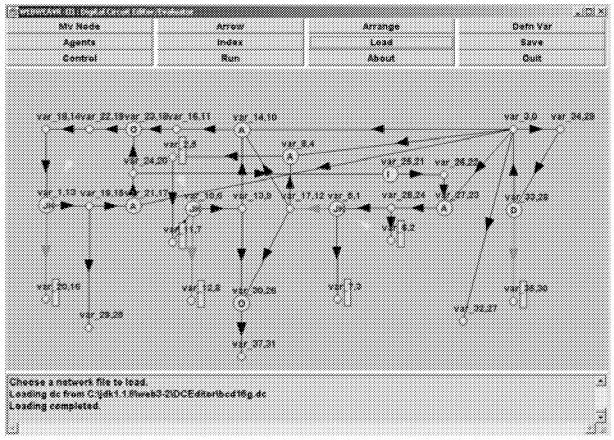


Figure 4: An evaluator and a loaded sequential digital circuit.

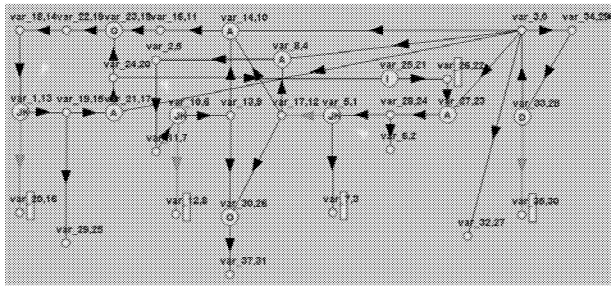


Figure 5: Variable values are updated based on the preceding state cycle by cycle.

The process is repeated cycle by cycle and hence all agents are synchronized with the simulator which provides dynamic environments.

6 CONCLUSION

We implemented a dynamic environment simulator which simulates the running of a sequential digital circuit with possibly faulty devices. The simulator allows multiple agents to observe the circuits simulated. Multiple agents are synchronized with the simulator through message passing. Experiments have been performed which show the simulator works well.

References

An, X. (2003). Probabilistic reasoning in dynamic domains under the cooperative multiagent paradigm. Technical report, School of Computer Science, University of Waterloo, Ontario, Canada.

Binder, J., Murphy, K., and Russell, S. (1997). Space-efficient inference in dynamic probabilistic networks. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 1292–1296, San Francisco, CA. Morgan Kaufmann Publishers.

Boyen, X. and Koller, D. (1998). Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-1998)*, pages 33–42, San Francisco, CA. Morgan Kaufmann Publishers.

Dean, T. and Kanazawa, K. (1988). Probabilistic temporal reasoning. In *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI-88), St. Paul, Minnesota*, pages 524–528. American Association for Artificial Intelligence.

Forbes, J., Huang, T., Kanazawa, K., and Russell, S. (1995). The BATmobile: Towards a Bayesian automated taxi. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*.

Jensen, F. V., Kjarulff, U., Olesen, K. G., and Pedersen, J. (1989). An expert system for control of waste water treatment - a pilot project. Technical report, Judex Datasystemer A/S, Aalborg, Denmark.

Kjarulff, U. (1992). A computational scheme for reasoning in dynamic probabilistic networks. In *Proceedings of Eighth Conference on Uncertainty in Artificial Intelligence (UAI-92)*. Stanford, California.

Kuenzer, A., Schlick, C., Ohmann, F., Schmidt, L., and Luczak, H. (2001). An empirical study of dynamic bayesian networks for user modeling. In *Proceedings of Workshop on Machine Learning for User Modeling*. Sonthofen, Germany.

Murphy, K. (2002). *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, Berkeley, California.

Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey 07458.

Xiang, Y. (2002). *Probabilistic Reasoning in Multiagent Systems: A Graphical Models Approach*. Cambridge University Press.

AUTHOR BIOGRAPHY

XIANGDONG AN was born in Tianjin, China. He received his B.Eng. and M.Eng. degrees from Tianjin University, China and Nankai University, China respectively. He is now studying at University of Waterloo, Canada for his Ph.D. degree in Computer Science.

A SIMULATION OF INTELLIGENT, KNOWLEDGE-BASED AGENTS EXPLORING THEIR ENVIRONMENT

Hubertus Franke
Hendrik Renken
Peter Scheideler
Andreas Schmidt
University of Paderborn
Fürstenallee 11
33102 Paderborn, Germany
E-mail: peter.scheideler@hni.upb.de

KEYWORDS

Search algorithms, constraint problems, Artificial intelligence, knowledge

ABSTRACT

Our scenario was set up in a very dynamic environment where autonomous shuttles must select their way from A to B according to a set of given constraints from a current passenger, orders coming in form other passengers wanting a lift from D to B and sensor input from the environment. To handle the amount of input data we separated the problems into two levels. On the first level the shuttle explores the most efficient path from point A to point B for one new passenger. Here we have the incoming orders from the passengers who like to travel within their defined constraints (e.g. price, time, additional passengers). On this level all possible paths which fulfil the constraints are selected. On the second level we face the problem to choose the most profitable and efficient way for a new passenger with already having a passenger onboard. So we must compare the new orders coming in with their constraints with the ones from the passenger we already have aboard. We will show a scenario and a method how to handle this complex scenario.

INTRODUCTION

Passengers express certain conditions on the way from A to B. These conditions impose special constraints on the shuttle. The relation between passenger and shuttle can be abstracted into a specific constraint problem notation where the passenger defines a certain amount of constraints for the shuttle. (1).

$$C \equiv \{s, g, t_0, \Delta t, c\} \quad (1)$$

s represents the start-station, g the end-station, t_0 the start-time, Δt the maximum period of time for the execution of the order and c the maximum of comfort. The more constraints the passenger defines the closer will become the solution space for the shuttle, meaning the passenger does not have to specify all.

The shuttle resides in a known state. This state is defined by the environment (e.g. place of shuttle) and the state of the shuttle itself (e.g. condition of the engine, battery). So the state (2) is a tuple of:

$$S \equiv \{ pos_0, \{f_1, f_2, \ldots\} s_s, t_a, typ\} \quad (2)$$

pos_0 is the actual position of the shuttle, $\{f_1, f_2, .. f_n\}$ is the number of loaded passengers, s_s is the actual status of the shuttle, t_a is the actual point in time, and typ is the type of the shuttle. A state is valid, if all constraints are met. If one constraint can not be fulfilled by the shuttle it would have to reject the order of the passenger
If the shuttle has classified its status as valid for the incoming order, it can now generate a number of actions (3), which transfer the existing status in a valid next status.

An action

$$A \equiv \{\Delta t, c, Z'\} \quad (3)$$

is defined by the following attributes:
Δt is the estimated time, which is necessary to perform the action, c are the costs which arise through reaching the next status Z'. It is now important for the shuttle to decide, whether the actions and which of the actions are profitable, relevant and efficient. This process (figure 1) is supported by the historic experiences the shuttle made in the past.

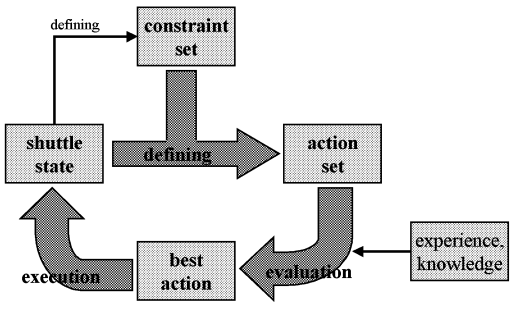


Figure 1: Selection cycle of a valid action (solution)

THE SELECTION CYCLE

The constraint problem

The passengers' constraints can be divided up into two groups: local and global ones. The local constraints integrate themselves disjunctively with the road network meaning that a local constraint formulates a special demand to each section of the net. One example could be the constraint of the passenger to only use high speed connections from A to B. According to this constraint all other ways loose their validity and are not allowed to be used by this shuttle, until the passenger was brought to its final destination.

Global constraints on the other hand integrate themselves with the previously used paths. One example is the time. If one passenger wants to travel within one hour from A to B, only those connections can be selected where the sum of the sections does not extend the one hour constraint.

Evaluation function and knowledge

The evaluation functions (4), (5) find the best (according to the constraints) way in a given road network. In this scenario the input data is the road network, the start and the destination. The method chosen here is a search algorithm, which evaluates the ways from the start to the final destination of the passenger. All necessary information is stored at the edges of the net. The edges do not only store theoretical or mathematical values given due to certain actual input data but also calculate conclusions out of historical data. Therefore a track frequently used by many shuttles and thus burdened with many traffic jams at certain times (e.g. in the morning or around midday) can be eliminated by the shuttle even though it might be the best way at the current time. In this scenario it is also possible for the shuttle to ask other shuttles for their advice. E.g. if one shuttle has never driven to Munich it can ask the other shuttles for their experiences. The method we use here is called Case-Based Reasoning (Bergmann 2002),(Beierle and Kern-Isberner 2003). This method matches the current situation with rated experiences made in the past. Our CBR-cycle is shown below.

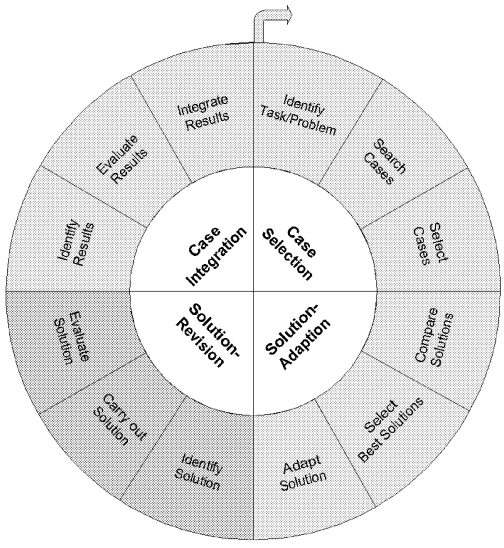


Figure 2: CBR-Cycle

Planning algorithm

If all information is know and the evaluation function has evaluated the possible tracks then the planning algorithm generates from a number of orders (o_1, ..., o_n), a set of actions (a_1, ..., a_n). An action could be a drive to a station where a passenger wants to enter a shuttle. The track to reach this station lies within the domain of the local optimisation problem.

A suitable order of actions could be:

At every time t the constraints c_1, ..., c_m of the orders o_1, ..., o_n must be fulfilled.

To solve this problem we use a sequential search algorithm (Russell and Norvig 2003), because it is the most efficient one to evaluate a next state based on a current state. Here an initial sub-solution is enlarged as long as a solution is found for the entire problem. The solution space is build up in a tree. The root consists of the initial state S_o which indicates the actual position of the shuttle. On the second level of the tree the nodal points consist of the initial state and a possible next state. The n-level consists of the initial state and the n-1 actions in all permutations.

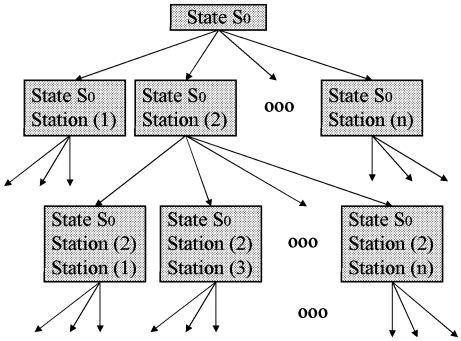


Figure 3: Structure of a search tree

The permutation of the n stations generates n! solution candidates. The problem is that n! solution candidates can not be efficiently searched. Therefore it is important to optimise and reduce the number of solution candidates. This can be done by an expansion of the nodal point n only by those successors which can be reached by the existing constraints. Additionally we need a good evaluation function that does not only estimate the costs but also the difficulties of the order (based on the constraints).

Such an integration of information is called an informed search and can be solved very well with the A*-algorithm (Russell and Norvig 2003), (Mitchell 1997). A* is a common algorithm that consists of two functions: g(n') calculates the costs for the previous way from the start to the actual nodal point n'; h(n') is an estimation function which estimates the costs from the actual nodal point to the next destination. An estimation function that always overestimates sometimes does not find an optimal solution. On the other side if the function underestimates it will always find the optimal solution.

The evaluation function, which searches out of the successors the most promising ones, should select stations which are positioned in the near area and can be reached with less cost.

Based on the requirements we generate the estimation function (4):

$$h(n) = \omega * \sqrt{\Delta x^2 + \Delta y^2} + (1-\omega) * c(st', st_n) \quad (4)$$

The first part of the equation calculates the distance from the last station to the next one. The part c(st', st_n) calculates the real costs to reach the next station. ω is the weight of the sub-functions and lies in the interval [0; 1]. With the weight it is possible to influence the actions of the shuttle.
The loading and unloading of the passenger has direct consequences on the road network, because each passenger represents a number of constraints. Therefore the possible drivable road network reduces with each new passenger. Therefore the search can very easily go into a dead end street if the shuttle tries to pick up a new passenger whose constraints reduce the road network so that the shuttle can not drive to any station without violating any constraints. To eliminate this problem the estimation function should be enlarged by a certain heuristic. Here the min/max-heuristic (Russell and Norvig 2003), (Stein 2002), (Schlette 2000) has shown excellent results in recent years. This heuristic tries to violate the constraints either as little as possible (min) or as much as possible (max). The first approach corresponds largely with the human brain. It tries to open as many solutions as possible and as long as possible. The max strategy tries to solve all big problems at the beginning and leaves the smaller and easier problems to the end.
Transferred to the scenario it means that the shuttle tries to unload the passengers first before it picks up new ones (Min-heuristic). The advantage is that the shuttle can always act on a large road network. In that way it is easier for the shuttle to react to sudden circumstances (traffic jam, accident). The disadvantage is that the passengers must pay more (if the shuttle picks up a new passenger the two passengers share the price of the trip). The max-heuristic turns the advantages and disadvantages of the min-heuristic into the opposite. However the risk to pay a retribution to one passenger, because the shuttle could not fulfil all his constraints, is high. Therefore the min-heuristic outweighs the max-heuristic. Therefore we need to enlarge our estimation function with C which stands for the number of constraints (5).

$$h(n) = \omega_1 * \sqrt{\Delta x^2 + \Delta y^2} + \omega_2 * c(st', st_n) + \omega_3 * |C| \quad (5)$$

Here the sum of the weights ω_i for i = 0..3 must be one.

Then the pseudo code of our planning algorithm is:

1. FOR EVERY c ∈ constraint(order):
    a. IF shuttleState breaks c THEN RETURN `FAILURE´.
2. Set position to shuttleState.Pos
3. Set constraints to shuttleState.Constraints
4. Save current state on STACK.
5. FOR EVERY o ∈ order(shuttle)
    a. IF start ∈ FINISHED AND goal ∉ FINISHED put goal -> NOTFINISHED
    b. ELSE IF start ∉ FINISHED put goal -> NOTFINISHED.
6. IF NOTFINISHED is empty RETURN FINISHED with `SUCCESS´
7. Set ReachableGraph <- ReduceGraph(pos, constraints).
8. FOR EVERY station s:
    a. IF s ∈ ReachableGraph AND s ∈ NOTFINISHED put s -> ACCESSIBLE.
9. IF ACCESSIBLE is empty get shuttleState from STACK AND GOTO 4. (Backtracking).
10. Remove from NOTFINISHED and place on FINISHED a station s for which potential is best.
11. SET pos <- s.
12. Update constraints due to NOTFINISHED.
13. GOTO 4.

The pseudo code of our ReduceGraph algorithm is:

1. FOR all edges ∈ Graph:
2. FOR all discrete Constraints:
3. IF edge NOT fulfil Constraint take edge out of Graph
4. AND GOTO 1.
5. FOR all outgoing edges ∈ Start:
6. FOR all addicted Constraints compute
7. markingDFS(Constraint, edge).
8. FOR all edges ∈ Graph:
9. IF edge NOT marked with every Constraint take edge out of
10. Graph.
11. remove mark in Graph.
12. RETURN Graph.

markingDFS:
1. Calculate Constraint.f(endVertex) FOR all Constraints.
2. IF 1 constraint is NOT fulfilled RETURN Failure.
3. IF endVertex is Goal RETURN success.
4. FOR all outgoing edges ∈ endVertex.
5. IF markingDFS(constraint, edge) RETURNS Success mark edge with constraint AND RETURN Success.
6. RETURN “false”.

ReduceGraph tests all edges if they violate the constraints and reduces the graph to the edges which belong to a valid path from the start to the destination. Here the constraints are divided up into two categories: the ones that are independent from the path (comfort, speed etc.) and the ones which depend on the current path (time, cost etc.).
First the discrete constraints are to be tested. This effects, that no paths exist which do not fulfil the constraints.

SIMULATION SYSTEM

Systems design

The system was designed under the consideration of speed, generic, easy and efficient tests of algorithms and communications between various shuttles to test communication protocols. Additionally the objects (shuttles) must be autonomous and capable to initiate actions independently so that a central system unit is not necessary.
We have built up a strong parallel and modular system (Kastens 2002), (Kastens 2003) so that new modules can be added easily to the existing system. Each object of the system is build upon this framework.

What we need, of course, is a communication server (figure 4) which exchanges the messages between the modules.

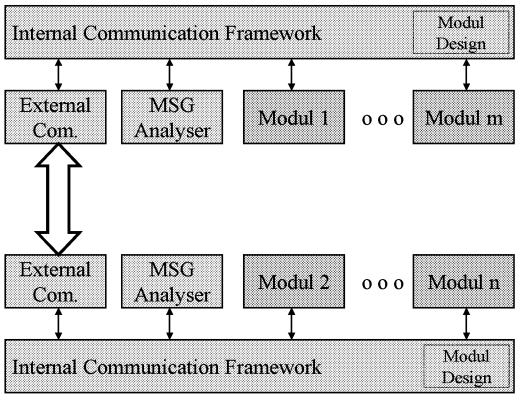


Figure 4: General Communication framework

Implementation

The components of the system are put into a hierarchical structure in which there are four categories: The group of the shuttles, the group of the fixed objects like the stations and the tracks, the plug-ins like the Log-Server or any kind of event generator and the group of the system services which are very important for the correct work of the system. They e.g. consist of the simulation engine or any kind of yellow pages.

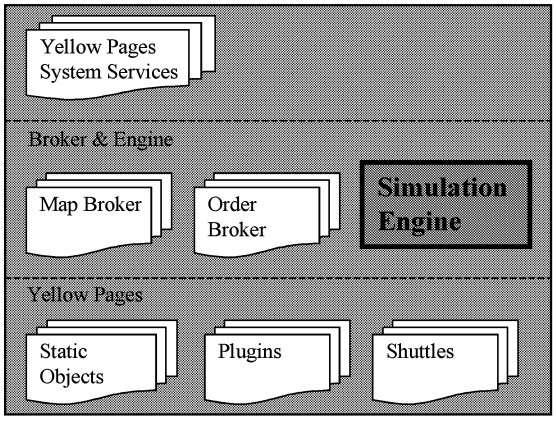


Figure 5: Part of the Simulation System

The pseudo code for the simulation engine is:

1. Compute For All Active Shuttle s Way in Actual Timewindow
    a. Lies Sensorpoint On Computed Way
        i. Set Shuttleposition To Sensorpoint
        ii. Ask Shuttle s What To Do Next
        iii. Wait For Answer
        iv. Compute Final Position From Answer
    b. Lies An Slower Shuttle On Computed Way
        i. Set Speed To Speed From Slower Shuttle
        ii. Inform Shuttle About Slower Shuttle
    c. Do Nothing If Nothing Lies On Computed Way
2. Return To 1. and Start With Next Time window

Example of the interaction of the shuttle

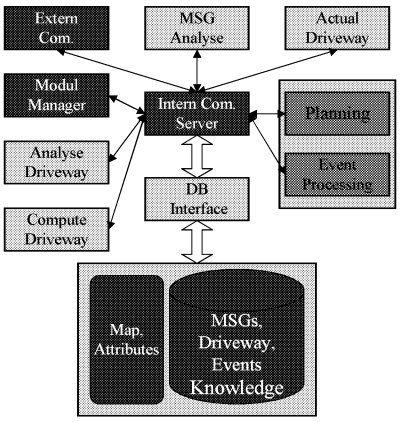


Figure 6: Shuttle module of the simulation engine

The "Internal Communication" Server, "module manager", "external communication" module and the "message analysis" belong to the object framework. The module manager is the module that starts, monitors and, if one module does not answer, restarts the module. The module "Actual Driveway" reacts to messages of the simulation engine, it answers, where the shuttle wants to drive to the next time. The planning module works through the incoming orders and fixes the path for the shuttle. It also reacts to unforeseen events like traffic jams, accidents with a change of the path in the "Actual Driveway" module. This uses the "Compute Driveway" to find the way from A to B. The learning components are in the module "Analyse Driveway". It compares the real data at the end of the trip with the calculated (expected) data. It so realizes problems which occurred during the trip and generates new knowledge out of them. Additionally it corrects the wrong weights of the edges.

Processing of an order

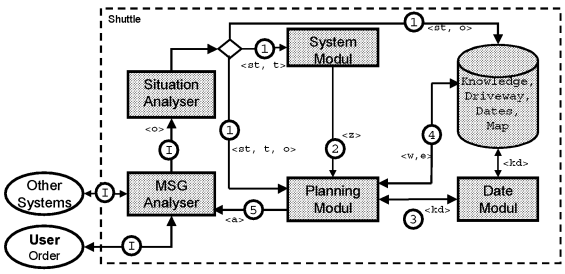


Figure 7: Framework of simulation system

The passenger generates at some kind of terminal an order which is send to an order broker which sends the order to all relevant shuttles. The "MSG Analyser" of the shuttle gets the message and knows that there is a passenger with an

order waiting for a response. The "MSG Analyser" checks the message if it is understandable and passes it on to the situation analyser. This module evaluates the order with the current status of the shuttle (loaded passengers, current position etc.) and passes the order on to the planning module and system module. The planning module tries to integrate the order into the actual path and actual strategy. If that is possible, the planning module calculates a price and sends it back to the MSG analyser. It then sends the offer to the terminal. If the passenger accepts the order the shuttle will start to calculate a new route in the driveway module.

CONCLUSION

We created a massive, parallel simulation through the approach of independent objects with a module structure. Through the separation of time in disjunctive intervals and the parallel processing by the shuttle objects, a calculation step only takes the time the slowest shuttle needs for its calculations.
With the development of the objects as modules it is possible to continuously improve the performance of the shuttle or to compare different kind of algorithms in different environments. We are currently developing a new module, which takes over the mentioned tasks but faster and more reliable. To integrate this new module it is sufficient to only exchange the relevant module without changing the architecture of the system. The simulation was designed in JAVA and with common database interfaces. Therefore it can be implemented on each system.

ACKNOWLEDGEMENT

This work was developed in the course of the Collaborative Research Center 614 – Self Optimizing Concepts and Structures in Mechanical Engineering – University of Paderborn, and was published on its behalf and funded by the Deutsche Forschungsgemeinschaft.

REFERENCES

Beierle, C. and Kern-Isberner, G. 2003. "Methoden wissensbasierter Systeme". Vieweg-Verlag, Wiesbaden 2003.

Bergmann, R. 2002. "Experience Management. Foundations, Development Methodology, and Internet-Based applications" Springer Verlag, Berlin, Heidelberg, 2002.

Kastens, U. 2002. "Introduction into objectoriented programming" In: Objektorientierte Programmierung. Lecture 2002/2003 http://ag-kastens.uni-paderborn.de/lehre/material/oop/

Kastens, U. 2003. "Parallel Programming, Introduction into Client-Server Principle". In Parallel Programming in Java. Lecture 2003 http://ag-kastens.uni-paderborn.de/lehre/material/ppje/

Mitchell, T. 1997 "Machine Learning", McGraw Hill, 1997.

Schlette, C. 2000. "Basics of Searchalgorithms". In: IRF Seminar 2000 Bahnplanung für autonome mobile Systeme http://www.irf.uni-dortmund.de/seminar/schlette/welcome.htm

Stein, B. 2002. "Enhanced Knowledge of Search algorithms". In: Wissensbasierte Systeme II: Suche Lecture 2002/2003 http://www.upb.de/cs/ag-klbue/courses/ws02/wbs2/search02-frame.ps.pdf

Russell, S. and Norvig, P. 2003 „Artificial Intelligence: A modern approach" Prentice Hall, Englewood Cliffs, N.J. 2003.

BIOGRAPHY

Peter Scheideler was born in Minden, Germany and went to the University of Paderborn, University of Illinois and University of Zaragoza where he studied business engineering and management. He obtained his degree in 1997. He worked for Procter & Gamble in the Customer Business Development department before he founded (with friends) a company which programmed software for the food industry. Currently he works for the University of Paderborn as a Ph.D. Student.

Hendrik Renken was born in Braunschweig, Germany and goes to the University of Paderborn since 2000. He studies computer science and supports the Heinz Nixdorf Institut in simulation and search algorithm projects.

Hubertus Franke was born in Höxter, Germany and went to the University of Paderborn, where he studied computer science. Since 1997 he was employed by the Bertelsmann AG as a software developer and technical project manager. Since 2000 he works for the Heinz Nixdorf Institute /University of Paderborn in Business Computing especially CIM and Agent Technology

Andreas Schmidt, born in 1970, currently works as a Research Assistant at the Computer Integrated Manufacturing Department of the Heinz Nixdorf Institute, University of Paderborn. He obtained Master degrees in Industrial Engineering at the University of Paderborn and International Management at the University of St. Thomas, Minneapolis, USA. After that he worked for 3.5 years as a business consultant and project director for a knowledge management company on several key-account projects in the banking, insurance and industry sector throughout Germany.

AI
IN
EXPERT
SYSTEMS

An Expert System for Yarn Dyeing

T. Hussain and M.R. Shamey
School of Textiles
Heriot-Watt University,
Scottish Borders Campus
Nether Road, Galashiels, TD1 3HF
T.Hussain@hw.ac.uk; r.shamey@hw.ac.uk

KEYWORDS
Expert System, Diagnosis, Textile yarn, Dyeing

ABSTRACT

This paper describes the development of a rule-based expert system for yarn dyeing. The system can be used as a diagnostic tool for troubleshooting problems in the dyeing of cotton yarn as well as for educational or training purposes. The system has been implemented in FuzzyCLIPS and the testing of the system has been done following the conventional expert system verification and validation methodologies.

INTRODUCTION

Textile diagnoses [Sampath, 2000; Smith, 1987; Radhakrishnan, 1984; Brent, 1984] can be distinguished into two main types: breadth-based and depth-based. A breadth-based diagnosis takes into account all the stages of textile manufacturing viz. fibre formation, yarn formation, dyeing and finishing. The main purpose of this type of diagnosis is to find out which manufacturing stage and characteristic feature of the product of that stage has been responsible for a particular defect. A depth-based diagnosis focuses on one manufacturing stage and takes into account all the parameters and factors of that stage that might have an effect on its final product. The main purpose of this type of diagnosis is to find out what error in parameters or fault in factors occurred that resulted in a particular defect [Tyndall, 97; Wiume, 83].

Defect formation in dyed yarn can be attributed to infinite number of causes ranging from poor quality of fibre, faulty spinning, inappropriate yarn package formation, impurities in water, poor standardisation of dyes and chemicals, machine malfunctions to human errors. Defect detection and diagnosis is essential in order to eliminate the root causes and hence improve the quality of dyed materials. The yarn dyeing expert system is capable of a comprehensive diagnosis, taking into account all the stages of textile manufacturing, and all the processing parameters to locate the root cause of a single or multiple defects in a dyed yarn. The system also gives the explanation how a particular cause may result in a certain defect and what preventative measures can be taken to avoid this cause in future dyeings.

THE DOMAIN KNOWLEDGE

Knowledge acquisition is seen as a crucial problem and has always been regarded as the bottleneck in developing any expert system. This bottleneck is mainly caused by communication difficulties between the knowledge engineer (KE) and the expert, the inability of the expert to describe expertise, and the inability of the KE to obtain expertise. The knowledge acquisition and system development for yarn dyeing expert system was done by a textile engineer, thus major difficulties in knowledge acquisition were avoided.

In the first phase, knowledge about the problems in the dyeing of cotton yarn and their possible causes was gathered through an extensive survey of textile journals, periodicals and books. In the second phase, this knowledge was consolidated by interviewing and sending questionnaires to four dyeing experts for eliciting their expertise about the dyeing problems and their possible causes as well as the diagnostic strategy employed by them. An example of the knowledge-base is given in Table 1.

Table 1: An Example of the Knowledge Base

Rule No.	If (defect/symptom)	Then (possible cause/ hypotheses)
1	Harsh handle	Silicate deposits High solid content in water Alkaline earth metals in water Too severe pre-treatment
2	Light-coloured areas/spots	Immature/dead cotton Formation of oxycellulose Silicate deposits Solid deposits Surfactants deposits Redeposition of impurities Hard water Broken fibres scattering more light Foreign fibres

THE DIAGNOSTIC KNOWLEDGE

There are three main sources of the diagnostic knowledge used in the system:
Inspection: symptoms (or defects)
Retrospection: product (or process) history

Examination: test data from the tests conducted on materials or process inputs
Symptoms (or defects) are the primary source of diagnostic knowledge that would initiate or trigger a diagnosis. These are the main causes of concern to the user which prompt the user for diagnosis. Without one or more of these, the initiation of the diagnosis is out of question. Product history is the account of the processes performed on the textile material and the process history is the state of the parameters during these processes. Last but not the least source of diagnostic knowledge is the data from the tests conducted on the dyed textile material and/or other process inputs such as water, etc.

Symptoms (or defects) are the most versatile type of knowledge which is utilized in diagnosis initiation, hypotheses generation, hypotheses sorting as well as hypotheses testing. The history and the test data are utilized, mainly, in testing of the hypotheses. As far as the domain knowledge is concerned, its main utilization is in hypotheses generation and hypotheses sorting. A summary of different types of knowledge and their utilization in the diagnostic expert system is given in Table 2.

Table 2: Utilization of Diagnostic and Domain Knowledge

Type of knowledge	Diagnostic steps where the knowledge is utilized		
Diagnostic knowledge			
Symptoms	Hypotheses generation	Hypotheses sorting	Hypotheses testing
History	-	-	Hypotheses testing
Diagnostic tests	Hypotheses generation	-	Hypotheses testing
Domain knowledge	Hypotheses generation		-
Certainty factors	-	Hypotheses sorting	-

SYSTEM DESIGN

Diagnostic expert systems can be developed either by using a general purpose programming language or an expert system shell. An expert system shell provides a substantial artificial intelligence code, including an inference engine that has been tested and debugged. It requires a shorter development time with less specialist programming experience. Although there are many expert system shells available, each has its own capabilities and limitations. Some important factors that influence the selection of an expert system shell include: knowledge capacity, knowledge representation and reasoning method, interfacability, source language, and money and time constraints.

To develop our system, we used FuzzyCLIPS. This expert system development tool represents knowledge as a set of IF–THEN rules (a knowledge representation approach well adapted to diagnostic expert systems). FuzzyCLIPS is characterized by forward chaining inference methods (although backward chaining is also possible), various confidence modes including fuzzy logic and the possibility to be integrated and to work with external programs. The user interface was developed by using wxFuzzyCLIPS, a tool for developing graphical user interfaces that allow to load and run FuzzyCLIPS applications.

An overview of the system is given in Figure 1. The input from the user includes information about symptoms or defects, product and/or process history, and test data. The output from the system includes the possible causes of the symptoms or defects, explanation of the diagnoses and corrective and/or preventative measures.

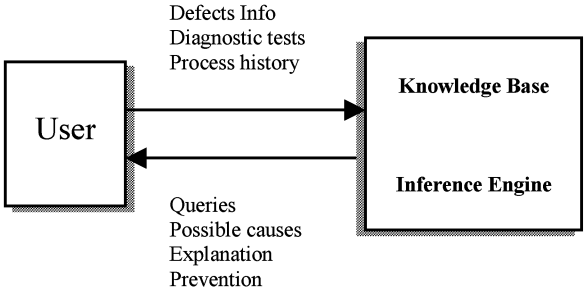


Figure 1 Inputs and Outputs of the System

The system is implemented in three modules (Figure 2). The diagnosis module helps the user in locating the most likely cause(s) of one ore more defects in the dyed yarn. The explanation modules, elucidates, how a particular cause (located by the system), can result in a particular defect and the prevention module gives the preventative measures.

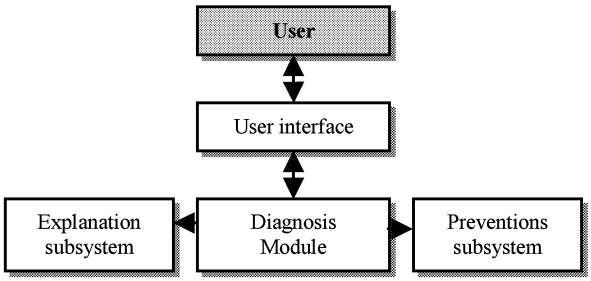


Figure 2 System Modules

DEALING WITH UNCERTAINTY

The list of possible causes of a defect manifest in a dyed material is usually very large. One of the main tasks in a diagnostic process is to decide in what manner and order these causes should be tested, for validation or refutation, to find the most likely or the root cause of the defect.

In deciding which of the possible causes of a defect should be tested first, no single criterion among chronology, classification, probability, certainty or commonness, is perfectly suitable. Testing/checking the possible causes in

chronological order do not guarantee minimum time, testing and effort. Classification is helpful in narrowing down the source of possible causes but some other criterion is still necessary to sort out the various causes in the shortened list. Causes can be tested/checked in the order of decreasing probability but requisite probabilistic data is neither available nor easy to implement. Certainty factors offer a good alternative, although they do not have as strong a theoretical basis as probability. So an appropriate combination of these criteria has been employed in tracing the causes of various dyeing problems.

INFERENCE PROCESS

The system is designed to perform diagnosis through abductive-deductive reasoning. When the user selects one or more symptoms or a defect, all possible causes of the selected symptom(s) or defect are asserted through abductive reasoning. Each possible cause has a different certainty factor, which is calculated as follows:

$$CFcause = CFsymptom \times Cfrule \qquad (1)$$

If more than one selected symptoms or defects have any cause in common, the final certainty factor of that cause is asserted as follows:

$$CFfinal = Max.\ (CFold\ ,\ CFnew) \qquad (2)$$

The certainty factor threshold for rule application is set by default to 0, so that all possible causes are considered no matter how low their associated certainty factors are. However, the user has the option to change the certainty factor threshold if the desire is to consider only the causes with high certainty (CF > 0.66) or only the causes with medium to high certainty (CF> 0.33).

The possible causes of the selected symptoms or defects, asserted through abduction, are stored in the temporary working memory of the system. These causes are then tested for validation or refutation by deductive reasoning, when the system solicits further input from the user in terms of product or process history and/or results of the tests performed on the defective dyed material. An overview of the inference process is given in Figure 2.

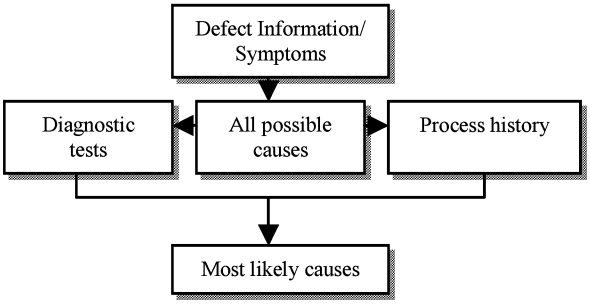


Figure 2 Overview of the Inference Process

USER INTERFACE

A screenshot of the user interface is given in Figure 3.

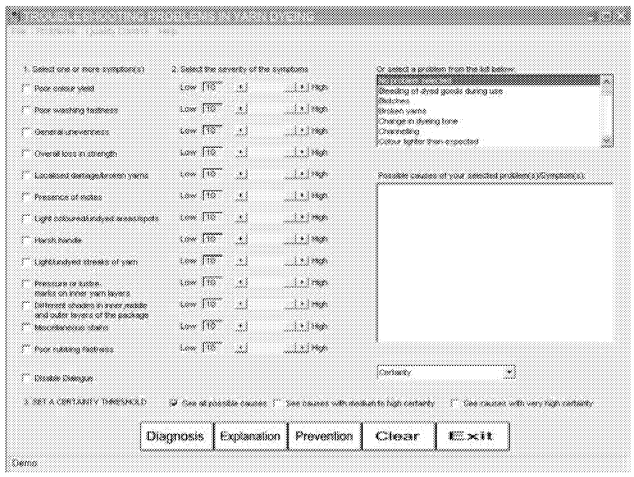


Figure 3 A Screenshot of the User Interface

The user can initiate the diagnosis in two different ways: by selecting one or more symptoms from the given symptoms or by selecting a defect from a given list of defects. The user has also the option for selecting the comparative severity of the symptoms present in the dyed material. After the user presses the "Diagnosis" button, the system comes up with all possible causes of the selected symptom(s) and/or defects. The user can investigate the causes only common among the selected symptoms or all the possible causes of the selected symptoms and/or defects. In order to test possible causes for validation or refutation, the system requests some more input from the user. It does so by asking the user questions about other symptoms that the user might have overlooked in his initial selection and questions about the processes performed on the textile material and/or the parameters used in these processes. The system can also ask the user to perform some tests on the dyed material and input the test results. On the basis of this new input from the user, the system comes up with the revised list of the possible causes, which it would give as an output to the user. The user can see the causes only with high certainty or causes with medium to high certainty or the complete list of revised/validated causes, by selecting different certainty thresholds. The revised possible causes given by the system are selectable. The user can select any of the possible cause to see the explanation how this cause can result in the defect in question and how this cause can be prevented.

VERIFICATION AND VALIDATION

The system was empirically tested after inspection and static verification. Faults were introduced deliberately during dyeing of yarn samples by a third person, who was not involved in the development of the system. The dyed samples were then given to the dyeing experts for the diagnosis. Two experts were chosen for the purpose. Expert A's experience was in fault diagnosis in dispute-settling cases, where the main objective is usually to establish

whether the fault is due to wet processing or processes prior to wet processing. Expert B's experience was in fault diagnosis for quality control, troubleshooting day-to-day problems in the pre-treatment and dyeing processes. The diagnosis was, then done with the help of the expert system. In the end diagnoses done by the human experts were compared against those done by the expert system. The comparison is given in Table 3.

Table 3: Comparison of the Performance of Human Experts and Computer Expert System

	Expert System	**Expert A**	**Expert B**
Total faults diagnosed	8	8	8
Total correct diagnoses	6	5	6
Total incorrect diagnoses	2	3	2
Accuracy %	75 %	62.5 %	75 %
Cause of incorrect diagnoses	Incorrect user input	Did not consider all possibilities	Did not consider all possibilities

As can be seen from the table, when the expert system was given the correct input, the accuracy was 100%. The system gave incorrect diagnosis only when it was given incorrect information. On the other hand, the human experts gave incorrect diagnoses in certain cases because they did not consider all possible causes of the faults, which the expert system did. Since the expert system contains the expertise of more than one expert, its knowledge base is more comprehensive as compared to individual human experts.

CONCLUSION AND FUTURE WORK

The results confirm the utility of the expert system for diagnosing dyeing faults. It can not only assist dyeing experts in finding the root cause of a defect but can help also novice dyers to diagnose common dyeing defects, on their own. The system is presently being expanded for diagnosing faults in woven and knitted cotton fabrics. In the future, it will be expanded further to other dye-fibre systems.

REFERENCES

Brendt H. J. 1984, "Faults in woven fabrics-causes, methods of reduction" *Textile Praxis International*, Volume 39, Number 3, Pages 238-241

Radhakrishnan, M. 1984, "Post-mortem of some confounding fabric defects" *Colourage*, Volume XXXI, Number 10, Pages 11-16

Sampath, M.R. 2000, "Frequently encountered problems in textile wet processing and a diagnostic approach for prevention/solution-Part I" *Colourage*, Volume XLVII, Number 9, Pages 73-75

Sampath, M. R. 2000, "Frequently encountered problems in textile wet processing and a diagnostic approach for prevention/solution-Part II" *Colourage*, Volume XLVII, Number 10, Pages 57-60

Smith, B. 1987, "Troubleshooting in wet processing- an overview" American Dyestuff Reporter, Volume 76, Number 2, Pages 28-30

Smith, B. 1987, "Troubleshooting in dyeing-Part 1: General" American Dyestuff Reporter, Volume 76, Number 3, Pages 17-26

Tyndall, M. R. 1997, "Quality aspects related to preparation, dyeing and finishing of cotton nits" American Dyestuff Reporter, Volume 86, Number 4, Pages 15-21

Wiume,W. 1983, "Systematic procedures for evaluation of defects in textile materials" Textile Praxis International, Volume 38, Number 10, Page 1071-1074

AN IMPLEMENTATION OF DYVELOP METHOD TO PROCESS SYSTEM DESIGN

Zbyněk Mikša, Zuzana Hošáková, Jana Dvořáková, Jaromír Dvořák and Jiří Urbánek
Brno University of Technology
Technická 2,
616 69 Brno, Czech Republic.
E-mail: urbanek@ust.fme.vutbr.cz

Veronika Urbánková
Masaryk´s University Brno
Komenského nám. 2
601 00 Brno, Czech Republic.
E-mail: vera.urban@email.cz

KEY WORDS
Algorithm, Design, Process System.

ABSTRACT

The **DYVELOP** (Dynamical Vector Logistics of Processes) method serves for the description, analysis, evaluation and management of Process Systems (PS) and their function and operation in the environments. Its implementation to PS Design asks a conversion of relativistic and logistic relation from heuristic to deterministic system. New methodology results in virtual model apparatus, which uses Venn's graph methodology in the PS Layout synthesis.

INTRODUCTION

The DYVELOP (Urbánek 1999) terms and relation are written using **CAPITAL BOLD FONTS** at first occurrence in next columns. Two Types of complicated PS are defined basically from anthroposophy (Urbánek and Skála 2001) viewpoints (see next Table).

Type PS	Specification
ORGANIZATIONS	An institution whose principal function isn't profit creation: tax-exempt; allowance, budgetary, governmental (schools, state and communal administration, military, police, fire brigades, courts, churches, state posts, etc.)

COMPANIES	Discriminative level is according to (e.g.):			
	operation	*output product*	*processes*	*industrial branch*
MANUFACTURING		**Semi-product**	Mining, Recycling, Processing, Fabrication,	Agriculture Chemistry Wood-working
		Final Product	Assemblage, Synthesis, Building, etc.	Machinery Electricity Architecture Food, etc.
SERVICE		Substantive	Reparation, Design, Logistics, Treatment,	Repair shops Editorships Transport Health
		Unsubstantial	Regeneration, Banking, etc.	Insurers Banks, etc.

A full address specification of the Type of PS is highly dependent on **discriminative level** (i.e. it is very relative). The decision about Type is a mission of strategic management. It designates principal aim also of future PS implicating a **PRINCIPAL FUNCTION** (Urbánek 2002). It expresses **OUTLET TERMINAL FUNCTION** at integrative relation to the **ENVIRONMENTS**. The tactical impletion of Principal Function requires the assessment of PS **TRANSIT FUNCTIONS** (i.e. **FUNDAMENTAL & SUBSIDIARY & POSSIBLE** functions). Than, the **Design** task results of tactical process-planning subdivision:

First, it is necessary to define PS Object (product) and analyse the Element (part) Basis of this Object. Feasible rate of the filling of object user's functions must be technologically revaluated parallel. Here, it is necessary proceed from the **Object Functions** to the **Functions of future PS**. On account of this, all functions of future PS must be planned out, which are necessitous for object creation. Further, the **Aims** an **OPERATION** (Urbánek, and Rýznar 2002) of the PS and an expectation of their time & extent filling must be defined essentially. It includes the **classification criteria specification** of future impletion of the Operational Aims, which must be determined at dependence to substantial surroundings & modification conditions & financing possibilities & environments. Then, a **PLAN** of Profile Structure of disposable technological resources & implements must be defined to PS Transit Functions filling. Further, a **PROGRAMME** of used Methods & Technologies is necessary to create (coherent with spatial-temporal structure of real part base of the object of the PS), by virtue of disposable **SOURCES** & expected trade **OUTLET** capacities. The necessity of capacitive requirements and/or possibilities harmonization results from of that.

METHODOLOGY

Above typical heuristic procedure presents the requirements and it gives evidence to the complication & severity of PS Design. This is a reason, we are concentrated further on only one type of the PS while we clear up PS **Design Procedures** – i.e. on **Manufacturing Company of machine-industries type with small-batch production**. The tactical exercise of **PROCESS APPROACH** (Hammer and Champy 1993) at above anthroposophy complicated Small-batch Production PS (SPPS) Design (Urbánek and Urbánek 2001) requires following steps:

A. A **Functional Analysis** of **PRINCIPAL FUNCTION** of future SPPS. The Principal Function implies from of SPPS anthroposophy type & aims & environments. It is the principal criteria function, whose satisfaction rate must be checked & optimised at all next steps. The reason is, that it is decisive (critical) function of future SPPS. The Principal Function definition is a result of this step as a Terminal (outlet) function for SPPS **environmental integration.**

B. The **Functional Analysis** (Dvořáková et al. 2002) of mostly characteristic **TRANSIT** (i.e. Fundamental,

Subsidiary, Possible) and **INLET TERMINAL** (i.e. Supplementary) **FUNCTIONS** of the SPPS. Integrative system of the **interactions & connections & relations** results from those functions, which will be taken into account at the SPPS Design. It is necessary to reconsider:

Ba. If the Design is influenced mainly by **a/ logistic** and/or **b/ relativistic** aspects -

Ba.a/ - *if leading role acts the **material flows** (key terms follows): manufacturing lines; capacity of hardware, machines, pipelines, tools, agents (factors); asked batch; timetable (scheduling), external-space connections, raw material source, fixed outlet; and so on.*

Ba.b/ - *if leading role acts the **technological relations**: the liaison, the information; agent environment, the structure dispensable anthroposophy surroundings (social, political, religious, ethnical, climatic, etc.); the relation to the source & agent & trade outlet environment; skill to changes; advanced demand on quality; and so on.*

The Design of the Manufacturing Company of machine-industries type with small-batch production (SPPS) is decisive influenced by ***technological relations*** mainly.

The SPPS fitting to Ba.b/ is a condition of Design routine:

C. A **PLAN** assigning the SPPS Design **methodical ENTITIES,** which follows from **Profile Structure** of disposable technological resources and implements - i.e. Processes, Technologies, Methods, Procedures, Operation, manufacturing instruments, tools, machines, production lines, human operators, handling devices, stores, traffic ways, package, etc. Those Entities must be potential able to fill the wide range of Terminal and Transit functions in requisite **PRODUCTIVITY**. It needs the **Capacity Calculations** of internal structure of methodical Entities. The quantitative and pragmatic Design of the means of the provision of the **Operations** is the result of this step.

D. An assignment of **Communication Manner & Control** among of above Entities and **MANAGEMENT Methods** (Urbánek 2001) of the Operation. The tendency is the maximal utilization of the computers, but only, if it brings more successful filling of the SPPS Principal Function. The informatics' and pragmatic Design of the **automation operational hardware & software** is the result of this step.

E. A heuristic **DECISION** about further pragmatic **Procedures** the SPPS Design, which must determine the decisive technology **Relations** for first Formalization Step. This step represents a conversion of SPPS Design from heuristic to determined system. The Decision depends on:

E.a *If the **"AGENCY RELATIONS"** are decisive, than a **Network Chart** as a graphic method of Operation Analysis can be used for the Agency Relation expression to advantage. This first Formalization Step uses The Graph Lines which represents & appreciates the Relation "Vehemence" between Nodal Points. The **Nodal Points** represent particular **PROCESS AGENCY**.*

E.b *If the **"KNOW-HOW RELATIONS"** are decisive, than the first Formalization Step proceeds at technology standardization implementation. The **Algorithms Development** represents **PROCESS TECHNOLOGY** as a conversion of SPPS Design from heuristic to determined systems in this case. The Algorithm of industrial process must be - elementary, determinative, aggregate, decisive and resultative. The result of this Formalization Step is the Algorithm of Manufacturing Process than.*

F. A **real PROCESS ENTITIES (e_k)** implements and allocates of **Operation Sectors** to the single steps of the Algorithms or of the Nodal Points. These Entities correspond with Fundamental, Subsidiary, Possible and Supplementary Process Functions fulfilling. The result of this implementation step is final **Manufacturing** or **Agency PROGRAMME.** The Manufacturing Programme formalises a technologic process optimisation at unified integrative data of systems structure – Object & Operation & Entity. The Agency Programme formalises the Agency Relation at integrative data of systems structure whereof the Object role is suppressed strongly. The Entities are modelled as the **Mock-ups** and/or **Areas** in SPPS **Layout**.

G. A creation of a **DIAGRAM** is important step of **LAYOUT DESIGN.** The Diagram contains Sub-diagrams:

G.a Relation's Sub-diagram as a Line Chart expresses the **RELATIONS** of the disjunctive (discreteness) Entities**.** Than, the simply conversion of the relations can represent **quantitative** (logistic) connections (ad Ba.a/) and/or **qualitative** relations (the Liaisons) (ad Ba.b/). Relation's Sub-diagram serves to the Relation conversion into simple **INTERACTIONS**. E.g. a **Virtual Interactions** can take into account a membership to identical Environments (e.g. family, set, group etc). Eventually, the Process **Productivity** and/or **Modality** can be represented as the Relation "Vehemence" by means Interaction-number multiplying between the Entities. The $\underline{i}_{xy}$ represents an Interaction count between elements (entities, environments, surroundings etc.) which are numbered **x, y**.

G.b Environment's Sub-diagram expresses the **Relations** of the **internal ENVIRONMENTS** (E) (Urbánek 1999). The Environment consists of an **entities group** that are able to do unary and/or binary mathematic operations [E; □]. Common **ATTRIBUTES** (Urbánek and Skála 2001) of the those environments can be: coincident characteristics; virtual parameters; profession organization of the entities (family ⟨group⟩ standardized technology); technologic indicia of process object (type standardized technology); handling device; areas articulation; pertinence to definite management; etc. Environmental membership can be expressed by means of above Virtual Interactions into the Relation's Sub-diagram (ad G.a). The Environment's Sub-diagram of **n**-process entities and **e**-environments inheres a mathematic matrix $\mathbf{E_{(n,e)}}$ type (n,e) with elements ε_{jk} increased of two values ⟨0,1⟩; (the 0 is unwritten). Necessary condition is (0) then

$$\mathbf{n \geq e} \qquad (0)$$

$$\varepsilon_{jk} = \{0,1\} \text{ for } j = I., II., III.,\ldots e.; \quad k = 1,2,\ldots n \qquad (1)$$

G.c Matrix's Sub-diagram $\mathbf{M_n}$ expresses the matrix of **entities relation occurrence**. From Relation's Sub-diagram (ad G.a) is possible to look up the number of interaction between couple entities. The number of interaction occurrence **x**-entity and/or **y**- entity of $\mathbf{M_n}$ matrix is marked $\underline{i}_{xy}$ and matrix elements give a values where the **N** is set of positive integers i.e {1,2,…n}

$$i_{xy} = \{1,2,\ldots \mu\} \quad \text{for } x,y = N \qquad (2)$$

It is evident, that interaction count $\underline{i}_{xy}$ generates square symmetrical matrix $\mathbf{M_n}$ type n,n with zero value of the elements on main diagonal, i.e. the Sub-diagram $\mathbf{M_n}$.

The analytical & evaluation phases are accomplished.

Heuristic pre-synthetic & synthetic phases follows:

H. The **pre-synthetic phase** involves the condition & rules definition of Layout Design. This step needs several sub-steps, the arrangement of they is articled in next regressive hierarchic sequence:

H.1 The definition of a **SURROUNDING (SUR)** of designed SPPS. It needs exact definition of the environment

of designed SPPS – a **0.E**. The both mathematic sets - the **SUR** and the **0.E** have a relation of disjunctive sets (they have not any conjunctive element). The rule is then

$$\mathbf{SUR} \notin \mathbf{0.E} \quad (3)$$

Then the negation $\mathbf{\overline{0.E}}$

$$\mathbf{\overline{0.E}} \equiv \mathbf{SUR} \quad (4)$$

Further is valid in the context of above term (1)

$$\mathbf{(I.E \cup II.E \cup \ldots \cup e.E) \subset 0.E} \quad (5)$$

E.g. peripheral walls of process area may represent the surrounding lines.

H.2 The Interactions $\underline{\mathbf{i}}_{\mathbf{xy}}$ must be specified specially among those entities $\mathbf{e_{k1}}$, which can be taken as a one-entity environment $\mathbf{j_{(1)}.E \equiv e_{k1}}$. Where

$$\mathbf{\Sigma e_{k1} = \Sigma j_{(1)}.E \leq n; \quad x,y = (1,2,\ldots j_{(1)}) ; x \neq y} \quad (6)$$

E.g. one-entity environment may represent the inspection workplace with one worker; buffer stock etc.

H.3 The **INTERACTIONS** $\underline{\mathbf{i}}_{\mathbf{xy}}$ must be specified among those entities $\mathbf{e_{kENV}}$, which appertain to particular environment **jp.ENV**.

$$\mathbf{e_{kENV} \supset jp.ENV} \quad (7)$$

E.g. particular environment may represent the turning shop, joinery, dispatch store etc.

H.4 The **INTERACTIONS** $\underline{\mathbf{i}}_{\mathbf{xy}}$ must be specified between entity $\mathbf{e_{k1}}$, which is one-entity environment **j1.ENV** and other environments **j.ENV**.

$$\mathbf{j1.ENV \equiv e_{k1} \supset j.ENV ;} \text{ for any } \mathbf{x, y} \quad (8)$$

E.g. the interaction between buffer stock and turning shop.

H.5 The **INTERACTIONS** $\underline{\mathbf{i}}_{\mathbf{xy}}$ must be specified among any of the environments **j.ENV**.

$$\mathbf{j.ENV \supset e_k} \quad (9)$$

E.g. the interaction among above in-nominated entities.

J. The **synthetic phase** of the **LAYOUT DESIGN** of future process system (SPPS). This step is pragmatic exercises as a graphic-mathematic method using methodology of Venn´s graphs (disjunctive sets with defined interfaces). First Layout model uses oval format of graphic representatives of the Entities. The area size of the Entities would be proportional to real size, which rises from above capacity calculations. The Process Entities are outlined with the **INTERFACES** importance. Firstly SPPS Layout model must be designed so, that the length of Interfaces of associated Entities must be proportional to their Interactions number. It must be recreated gradually to the final layout form, which beside above conditions respect their real shape and format (Mock-ups, Areas).

J.a The Entity $\mathbf{e_{kmax}}$ with maximum number of Interactions $\mathbf{max}\ \underline{\mathbf{i}}_{\mathbf{xy}}$ is designed to the centre of the first SPPS model. The oval shape of this set must respect real format (e.g. transport way must be oblong).

J.b Further associated sets (Entities) are "glued" to the $\mathbf{e_{kmax}}$ without any conjunctions and/or "white-space" in the order of their occurrence. The length of the Interfaces of further associated Entities must be proportional to interaction numbers. The Entities are outlined abreast appertains to particular Environment. Compact configuration of the sets is desirable.

J.c If the other associated entity (set) is not possible to "glue closely" to related entities, than it is necessary place it to the closest neighbourhood. The Entities "among" would fill Subsidiary or Possible functions only. The length, shape and neighbourhood of Interfaces have great evidence capability in designing models.

CONCLUSION

The formal apparatus of DYVELOP method implementation has been used for Process System Design at Manufacturing Company of machine-industries type with small-batch production. Practical exploitation will be stated on the talk in course session. It will include the creation of the model of Process System utilizing user's routinely accessible MS Word software.

REFERENCES

Dvořáková,J.; Mikša,Z. and Urbánek,J.F. 2002 "Theory of Processes – application possibilities", *The 13th INTERNATIONAL DAAAM SYMPOSIUM , "Intelligent Manufacturing & Automation: Learning from Nature"*, Wien, 193-194, ISBN 3-901509-29-1

Hammer,M. and Champy,J. 1993. "Reengineering the Corporation: a Manifesto for Business Revolution". *Harper Colins Pb.,Inc. USA.*

Rýznar,B. and Urbánek, J.F. 2002. "Combat Simulation Terminology Shift to Process Crisis Management". *MESM´2002 International Conference*, Sharjah, U.A.E., pp. 73-78, ISBN 90-77039-09-0

Urbánek, J.F. 1999. "The Recycling of Waste on Multi-grade PLC Structure", *1st International Conference on Solid Waste, Technology, Safety, Environment,* Rome, Italy, ICSW –99-1803.

Urbánek, J.F. 1999. "New Instrument of Integrated Waste Management–DYVELOP", *15th International Conference on Solid Waste Technology and Management,* Philadelphia, U.S.A., ISSN 1091-8043

Urbánek, J.F. and Urbánek,K. 2001. "Environmental Process Design & Theory of Processes *". VR-Mech´ 01, Royal Military Academy,* Brussels, Belgium.

Urbánek,J.F. and Skála,Z. 2001. "Theory of Processes© - new Language of Human Sustainable Development", *International Containment & Remediation Technology Conference and Exhibition,* USA, Orlando.

Urbánek, J.F. 2002. "Theory of Processes - Management of Environments*". Czech language book,* CERM Brno, ISBN 80-7204-232-7.

Urbánek, J.F. and Rýznar,B. 2002. "Environmental Theory of Processes", *MESM´2002 International Conference*, Sharjah, U.A.E.,. pp. 209-213, ISBN 90-77039-09-0.

Urbánek,J.F. and Urbánek,K. 2003 "The Theory of Process Relativity upgrades the Systems Concepts". *Symposium of Critical Design Issues for the Human-machine Interface*, NATO Conference, Prague

AUTHOR BIOGRAPHY

Zbyněk Mikša was born in Brno, Czech Republic and went to the Masaryk´s University, Faculty of Informatics, where obtained his degree in 2002. Now, he is doctoral student of Brno University of Technology and he solves the tasks of process management and its computer aid.

THE CVCA MODEL: A CELLULAR AUTOMATON MODEL OF LANDSCAPE ECOLOGICAL STRATEGIES

Jack Wileden
Computer Science Department
University of Massachusetts
Amherst, MA 01002, USA

Email: jack@cs.umass.edu

Elisabete A. Silva
CESUR
Lisbon Technical University
Av. Rovisco Pais
1049-001 Lisboa, Portugal
Email: elisabetesilva@ist.utl.pt

Jack Ahern
LARP
University of Massachusetts
Amherst, MA 01002, USA

Email: jfa@larp.umass.edu

KEYWORDS
Landscape Ecological Strategies, Cellular Automata, Lisbon, Porto, Portugal

ABSTRACT

The application of landscape ecological strategies to the urban growth of two metropolitan areas is detailed in this paper by developing a cellular automata model coupled with an existent urban model. The implementation of the cellular automaton model CVCA (Countervailance Cellular Automata) loosely coupled with SLEUTH (a CA urban model) using two case studies provides a robust and useful application of landscape ecological strategies for metropolitan planning. These strategies vary locally as a function of the specifics about particular patterns and processes to be promoted and local characteristics. This paper makes an overview of the different phases of CVCA development and implementation.

INTRODUCTION

The need to include landscape ecological strategies in metropolitan planning is one of the starting points of this research. The main challenge was to develop a model that includes robust theoretical considerations and that is capable of being sensitive to the local conditions. Cellular Automata are capable of answering both needs. The capacity of these models to simulate local behavior according to a set of rules of neighborhood, several cell states, and time constraints are being recognized as valuable elements when planning complex systems while simultaneously guaranteeing portability and adjustability to local characteristics.

The CVCA model was conceived to provide answers to questions such as: What is the state of the landscape? Which landscape strategies are more predominant?, What is the dominant pattern (corridor, patch)? Are the strategies promoting connectivity? Which landscape metrics increase a dominant strategy? In order to answer these questions several components of the CVCA model had to be defined, including: landscape metrics, decision rules, and interaction with the urban model.

This paper makes an overview of the theoretical considerations, the options for specific metrics that best describe the system, the decision rules developed in order to answer to questions listed that the model should answer, and the capabilities and opportunities of running CVCA and SLEUTH loosely coupled with landscape ecological strategies included.

THE MODELS CVCA & SLEUTH

The Countervalance Cellular Automata (CVCA) is a loose coupled program designed to work in conjunction with SLEUTH (an urban cellular automata simulation model). CVCA operationalizes a set of landscape ecological strategies for urban planning. CVCA first assesses a landscape against a set of landscape metrics, then evaluates the future allocated urban cells from SLEUTH against the metrics and allocates future land uses according to a suite of planning strategies (offensive, defensive, protective, opportunistic). The simulations of future metropolitan land use patterns resulting from the interaction of CVCA and SLEUTH identifies ecologically important corridors and buffer areas around patches of high ecological value. This paper describes the development of CVCA, the theory behind the integral landscape strategies, and the actions taken to bring CVCA into a computable environment. An application of CVCA in two metropolitan areas in Portugal (Porto and Lisbon) is made and discussed.

SLEUTH is the acronym for Slope, Land Use, Excluded Areas of Urbanization, Urbanization, Transportation, and Hillshade (the input layers that will make the model run). SLEUTH requires five input images: urbanization, transportation, excluded areas from urbanization, slopes, and hillshade. The images for each layer are prepared

using GIS software and then converted to 8-bit gif format and stored in separate files. In SLEUTH each cell in a layer receives a value: 0 is a null value, while all the values between 1 and 255 represent values appropriate to the layer (e.g., extent of urbanization)..

Through analysis of the files SLEUTH mimics several types/forms of urban growth, including: organic growth, road-influenced growth, and diffusive growth. Cell evolution of urbanization is recorded and supports a set of self-modification rules in order to control the parameters, allowing the model to modify itself and therefore making the resulting simulations closer to what is observed in the real world. For example, the detection of intense growth periods (boom) or periods of little or no growth (bust), changes the control parameters.

The results from running SLEUTH include year-by-year images of urban growth throughout a metropolitan study area. These images contain: the initially existing urban cells, and cells with different probabilities of growth (from high to little probability of growth), these cells are placed in the background (hillshade file).

The CVCA model requires the same input layers as SLEUTH (Slope, Hillshade, Transportation, Urban, Excluded). The excluded layer had to be changed significantly, however in order to distinguish a different class that represents all the areas outside the boundary of the metropolitan areas. These areas are excluded from urbanization but for CVCA purposes they cannot be considered the same type of exclusion, since they do not figure in the application of the ecological strategies.

The output files of both models are the same, although the year-by-year output images are somewhat different. SLEUTH outputs different classes against a hillshade background: existent urban, simulated urban with high probability of change, or simulated urban with lower probabilities of change (this class has several subclasses of probability, from very low probability to average-high probability). The CVCA outputs four more classes that match four different landscape planning strategies. As discussed earlier, it is important to develop this kind of model, where urban and environmental dynamics can be integrated and sensitivity to local conditions is addressed.

THE CVCA ALGORITHM

The CVCA algorithm is based on the research developed by Jack Ahern (1998) that proposes an alternative future scenario analysis based in a set of essential attributes of a Landscape Ecological Planning Model. Ahern defines the Landscape Ecological Panning Model as an interdisciplinary approach that addresses landscape pattern:process at multiple scales, and includes a "human ecological component" (Ahern 1998: 4).

Ahern's Framework Method (1998) for landscape ecological planning focuses on a set of possible, non-mutually exclusive planning strategies: 1. Protective Strategy (when the existing landscape supports the abiotic, biotic, and cultural -"abc"- resource goals), 2. Defensive strategy (when the existing landscape is already in a spatial configuration that is negatively impacting "abc" resources), 3. Offensive strategy (adopting a proactive action when the landscape is already deficient with respect to supporting "abc" resources), and 4. Opportunistic Strategy (since very frequently landscapes contain unique elements or configurations of elements that represent positive opportunities) (Figure 1).

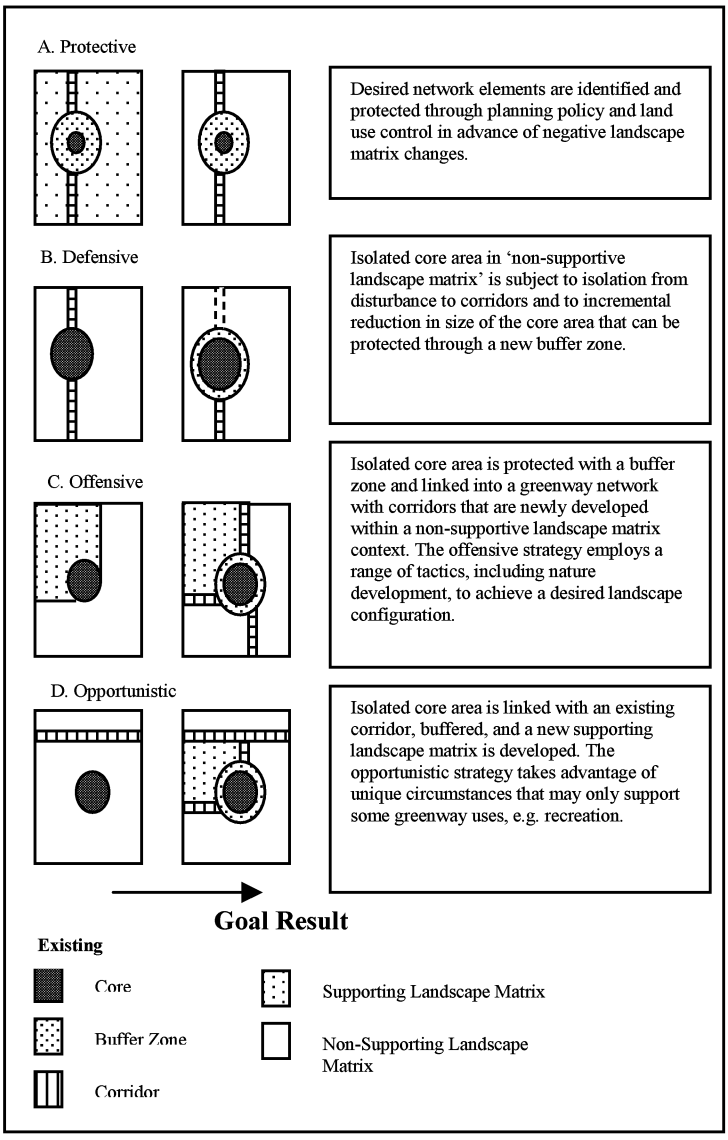


Figure 1: the Landscape Ecological Strategies developed by Ahern (1998)

Some adjustments needed to be made in order to translate Ahern's landscape strategies into landscape metrics and

then to develop an appropriate code that the CVCA model could run (Table 1).

Following cellular automata concepts, CVCA is defined by a grid space (the urban map and the map with areas of natural interest), by a neighborhood effect (8 cells neighborhood), and by a set of transition rules (protective, defensive, offensive, opportunistic, and let it grow), and a time step iteration.

The time step iteration works according to a sequenced time step (in this case a year-by-year basis). It begins with a seed year urban layer (e.g. 2002), looks at the cells with some probability of change and applies the five landscape strategies, according to the neighborhood effect, and the landscape metric MNND, and the different probabilities of change.

The Mean Nearest Neighborhood Distance (MNND), is one of several metrics that the model needs to compute in order to produce results.

The metrics used were chosen by the spatial indicators that have been mentioned in different landscape ecological studies (MacGarigal and Marks 1995; Gustafson 1998; Turner 1991). They are, the Landscape Shape Index (LSI), the Mean Patch Size (MPS), and the Mean Nearest Neighbor Distance (MNND). Their description is as follows:

Landscape Shape Index

Units: None
Range: LSI>= 1, without limit
LSI=1 when the landscape consists of a single circular (vector) or square (raster) patch; LSI increases without limit as landscape shape becomes more irregular or as the length of edge within the landscape increases, or both.

Mean Patch Size

$$MPS = \frac{A}{N}\left(\frac{1}{10{,}000}\right)$$

Range: MPS>0, without limit
MPS= Mean Patch Size; A= total landscape area (in cells); N= total number of patches

Mean Nearest Neighbor Distance

$$MNND = \frac{\sum_{j=1}^{n'} hij}{n'j}$$

Range: MNND>0, without limit
MNND=Mean Nearest Neighbor; h=distance; i=patch; n=number of patches; j=neighbor

Table 1: The transition rules at CVCA

Transition Rules:	Number of cells (cells with a probability of change to urban)	Action step
Offensive:	>50%	add offensive cells to all outer patch cells and add offensive cells corridor to nearest neighbor
Defensive:	<=50% *,**	add defensive cells to all outer patch cells where transition cell exists
Opportunistic:	0 but NN <= MNND	(and no transition cells adjacent) add opportunistic cells corridor to nearest neighbor
Protective:	0 but NN > MNND	add protective cells around all outer patch cells and add protective cells corridor to nearest neighbor
5. Grow: CVCA leaves the cells that are not subject to strategies. These cells feed SLEUTH again are the base for the next simulation		
* if in this 50% of cells more than half are high probability of change then add defensive cells to all other patch. ** if in this 50% of cells more than half are high probability of change, and NN is half of MNND than add defensive cells to all other patch and connect to NN.		

The Mean Nearest Neighbor Distance (MNND) has an important function in the opportunistic and in the protective strategy. The precondition for applying the opportunistic strategy does not require an urban pixel to be adjacent to any excluded areas of urbanization. The only requirement is that the distance to the Nearest Neighbor (NN) needs to be less than or equal to the Mean Nearest Neighbor Distance (MNND); in this case the strategy will be to establish a corridor to the nearest neighboring patch.

Finally, the environmental countervailing code works as specified in the pseudocode given in Table 2. One of the key actions of the CVCA model is the computation of the landscape metrics, and the development of the strategies. Because there is only “excluded” data for one year (SLEUTH just requires one excluded year, and also there is no digital data in Portugal for more than one excluded year), the model receives this excluded data while simulating the first year, uses it to compute the landscape metrics, and constrains the future evolution of the system by permanently defining where the excluded areas are.

Besides SLEUTH, Fragstats(1) was used as a reference system, mainly in terms of the data structures used in order to identify the proximity of different patches and the patch itself. While Fragstats was not used as the main software, there is Fragstats code in the program (the code calls and runs the landscape metrics against it). This code integration is very important not only to optimize the reading of this landscape metric, but also for further development in case other metrics are to be included in further developments of CVCA.

Table 3: Pseudocode of CVCA

Pseudocode	Comments
Initialize SLEUTH (includes marking excluded cells);	/* Read urban, roads, slope, exluded,
Initialize CVCA (includes computing metrics – MNND, LSI, MPS – and building lists of excluded areas [patches] and their boundary cells and distance to their nearest neighbors [NN]);	hillshade data*/ /* Metrics computed using modified fragstats */
Begin SLEUTH simulation of urban growth;	/* Patches based on modified excluded data */
For each SLEUTH iteration (each simulated year)	/* Nearest neighbors found via modified
For each patch (excluded area) P	fragstats */
P.incursions = number of boundary cells of P that SLEUTH proposes for urbanization	
If P.incursions == 0	
If MNND/2 < P.NN <= MNND then P. strat = 0;	
Else if P.NN <= MNND/2 then P.strat = 1;	/* Choice of Landscape Strategies based
Else if P.NN > MNND then P.strat = 2;	on:*/
Else if P.incursions <= 50% of P boundary	/* 1) Extent of incursion on patch
If P.NN <= MNND/2 then P.strat = 3;	boundary */
Else if P.NN > MNND/2 then P.strat = 4;	/* 2) Proximity of nearest
Else if P.incursions > 50% of P.boundary	neighboring patch */
then P.strat = 6;	/* P.strat == 0 means allow growth */
Produce output map based on P.strat	/* P.strat == 1 means opportunistic
P.strat ==1: Add opportunistic corridor to nearest neighboring patch;	strategy */ /* P.strat == 2 means protective strategy */
P.strat ==2: Add protective border to patch and opportunistic corridor to nearest neighboring patch;	/* P.strat == 3 means defensive strategy */
P.strat ==3: Make boundary cells that SLEUTH proposed for urban Into defensive border cells;	/* P.strat == 4 means more defensive strategy */
P.strat ==4: Add defensive border to patch;	/* P.strat == 5 means even more
P.strat ==5: Add defensive border to patch and defensive corridor to nearest neighboring patch;	defensive strategy */ /* (P.strat==5 not implemented in current CVCA) */
P.strat ==6: Add offensive border to patch and offensive corridor to nearest neighboring patch;	/* P.strat == 6 means offensive strategy */
Modify proposed urbanizations for SLEUTH by deleting all proposed urbanizations in patch boundary and corridor cells;	/* Modified SLEUTH input map for next iteration */

The results of applying CVCA loosely coupled with SLEUTH can be both qualitative (through the output of maps) and quantitative (through the output of tables). The maps have information that can be used in two different ways: 1.feed SLEUTH an urban version that just indicates where urban growth has been permitted, but does not explicitly show where the landscape strategies were applied; 2.display the results of applying the landscape strategies.

CONCLUSIONS

The development and implementation of CVCA demonstrates the feasibility of developing integrated models of urban and environmental characteristics and being at the same time sensitive to local characteristics.

Prior to the development of CVCA, SLEUTH was successfully run with two case study metropolitan areas in Portugal (Silva and Clarke 2002). After CVCA was implemented, it was applied to these same two case study metropolitan areas and used to show how applying the ecological strategies might alter the course of urban growth.

Despite the successful implementation and application of CVCA, further development and research remain to be done. For example, one important issue concerns the existence of barriers to the establishment of corridors. The protective, opportunistic and offensive landscape strategies imply the creation of corridors. The "excluded" maps of ecological and agricultural reserves do not include existing urban areas (or transportation) and when simulating urban growth and imposing the strategies at the same time, frequently a strategy creates a corridor that 'cuts across' an existing urban area. This poses a potential problem, since it is questionable whether we can 'cut across' these existing human structures each time the nearest neighbor computation in the model dictates cutting across an existent urban area. In this initial version it was accepted that CVCA-generated corridors might cut existent urban areas. In reality this would imply an ecological restoration, indicated as an offensive strategy (Ahern 1998).

AKNOWLEDGMENTS

This research was supported by two grants: between 1998 and 1999 from the Luso-American Foundation (FLAD); and between January 2000 to August 2002, from the Portuguese National Science Foundation (FCT-PRAXIS XXI). The University of Massachusetts Amherst also provided support for Dr. Silva's dissertation research.

REFERENCES

Ahern, J. 1998. "Spatial Concepts, planning strategies and future scenarios: a framework method for integrating landscape ecology and landscape planning". In J. Kopatek & R. Gardner, Landscape Ecological Analysis: Issues and Applications (pp.175-201). New York: Springer-Verlag Inc.

Gustafson, E. 1998. "Quantifying Landscape Spatial Pattern: What is the State of the Art?", Ecosystems, 1, 143-156.

McGarigal, K. and Marks B. 1995. "FRAGSTATS: spatial pattern analysis program for quantifying landscape structure". USDA For. Serv. Gen. Tech. Rep. PNW-351.

Silva, E., and Clarke, K. 2002. "Calibration of the SLEUTH urban growth model for Lisbon and Porto, Portugal" Computers, Environment and Urban Systems, 26 (6): 525-552

Turner, M., and Gardner, R. 1991. "Quantitative Methods in Landscape Ecology". New York: Springer.
(1)Fragstats was developed by Professors MacGarigal and Marks, at UMASS

NEURO FUZZY MODELLING

DATUM PLANE'S COVER AND MEMBERSHIP FUNCTIONS IN FUZZY MODELLING

Benmakrouha Farida [1]

[1] Computer Science Department INSA - IRISA

20 Avenue Buttes de Coësmes,CS 1431 35043 Rennes Cedex

Abstract

In the last years, Takagi-Sugeno fuzzy models have been used efficiently for the approximation of nonlinear systems. Many papers propound algorithms for extraction of knowledge from numerical data. But, few works have been developed for design of experiments and datum plane's cover. However, these points are crucial in fuzzy modelling [1, 2].
This paper tries to measure the impact of datum plane's cover on the outcome of a fuzzy inference system. In [1], the author propose a criterion which is optimal coverage of datum plane by a set of experiments. We analyse and compare it with another measure proposed in this paper.
We show that performance is not necessarily improved with training or number of rules. We propound, using these measures, a pre-validation of a fuzzy model. In fact, we show that a fuzzy inference system can't be suitable for describing the real system on some conditions.

1 Introduction

In a zero-order Takagi-Sugeno model (T-S model)[3], the rule basis are of the form :

$$\textbf{If } x_1 \text{ is } A_{1i} \textbf{ and } x_2 \text{ is } A_{2i} ... \textbf{and } x_p \text{ is } A_{pi} \quad \textbf{then } y \text{ is } C_i \tag{1}$$

where x_j are input variables, A_{ij} are linguistic labels on the universe of discourse X represented by a fuzzy set $\mu_{A_{ij}}$, y is the inferred output and C_i are constant values. The output y is, given an input vector x_1,...,x_n:

$$y(x) = \frac{\sum_{i=1}^{N} \alpha_i(x) \times C_i}{\sum_{j=1}^{N} \alpha_j(x)} \tag{2}$$

where the truth value of rule i $\alpha_i(x) = \mu_{A_{i1}} ... \mu_{A_{iN}}$ and N the number of rules.
The conclusions C_i can be initialized by an algorithm described in [4]. But, when data don't cover the whole input space, some rules are never learned and methods for determining conclusions don't work very well.
The structure of the paper is as follows : In the second section, we analyse and compare two measures of datum plane's coverage. We propound, using these measures, a pre-validation of the fuzzy model.
In section 3, we emphasize, with three examples, the importance of these criteria on the performance of a fuzzy inference system.

2 Pre-validation of a fuzzy system

We suppose that there exists a learning set $\mathcal{L} = \{(\mathbf{x}_i, d_i)\}$, where $\mathbf{x}_i$ is an input vector and d_i, the corresponding output.
Usually, to validate a fuzzy inference system, the mean square error (MSE) is calculated on a test set. If the MSE exceeds a threshold, then training is made, using a gradient method. This consists in modifying C_i at each presentation of examples from the error $(y(\mathbf{x}_i) - d_i)$.
Unfortunately, in case of model's invalidation, we can't determine rules never learned that cause the gap between the model and the real system. Moreover, if there is an insufficient coverage of datum plane, training and finer splitting of input space are inefficient and useless.
With these two criteria proposed below, we estimate the datum plane's coverage and we are able to isolate unactivated rules. Then, partial remodelling of the fuzzy inference system is possible.

2.1 First criterion :

In [1], the author propose a criterion named r_i which measure datum plane's coverage by a set of experiments. The idea is to split datum plane into fuzzy sets and to compute an absolute level of realization r_i :

$$r_i = \sup_{(P_k \in P)} \inf_{(Q_{ij} \in E_i)} \sup_{(x \in D_j)} \mu_{Q_{ij}}(x_j) \tag{3}$$

P_k are the set of experiments.
E_i is a set of labels Q_{ij}
Q_{ij} are linguistic labels represented by a fuzzy set $\mu_{Q_{ij}}$(according to a factor F_j) .
Each F_j has a value x_j, variable in an experimental set D_j

If this absolute level of realization is above a threshold, the experimenter consider that datum plane is sufficiently covered, otherwise, it's not and additional experiments are due, for improving coverage of concerned area.

2.2 Second criterion :

We define the support of fuzzy set A_i on T the test set

$$S_{A_i} = \{x \in T | \alpha_i(x) > 0\}$$

We compute the mean, for each rule i (m =card(S_{A_i})):

$$\beta_i = \frac{\sum_{j=1}^{j=m} \alpha_i(x_j)}{m}$$
$$\beta = inf_i(\beta_i)$$

If this value β is above a threshold, the model is prevalidated, otherwise it's not.

2.3 Comparison of these two measures

We recall steps of fuzzy modelling :

- **Step 1** : Experiments
- **Step 2** : Extraction of knowledge from data
- **Step 3** : Optimization of the FIS
- **Step 4** : Model validation

The first method takes place early in fuzzy modelling, just after experiments (for the second method, it's before step 4). The advantage is to make some more tests and improve coverage, if necessary, before going on. But it supposes that a datum plane is pre-defined before data analysis, that's not always verified.
Moreover, in the first method, fuzzy model is prevalidated even when data are scarce (but well shared).
We show this in the following example.
We consider the function $y(t) = \frac{sin(t)}{t}$.
The training set (resp the test set)is the interval [0.2 9.0](resp [0.2 11.0]).
The T-S model is composed by n rules.
We take $r = \inf_i(r_i)$

n	MSE	criter 1 : r	criter 2 : β
3	0.06	0.98	0.27
7	0.06	0.96	0.22
10	0.04	0.95	0.48

Table 1: First Table .

We remark that the absolute level of realisation is very good (near 1.0), while validation gives poor results.

3 Case examples

We compare, using three examples, output inferred by fuzzy systems :

- when datum plane is "sufficiently" covered
- when datum plane is not well covered
- when datum plane is not covered

The mean squarre error, performance index of the approximate system, is calculated on measures of the test set, before optimization step.

3.1 First Example

This example has been shown in [5].

$$y = \frac{sin(x)}{x}$$

In this paper, the authors propose a method to improve Takagi-Sugeno' approximation, under assumption that datum plane is covered. If this assumption no longer holds, we see that this method can't work, since it implies that, before optimization, the fuzzy system gives acceptable results. We show that it's not the case if training and test sets are disjoined.

Dat pla	Test intl	MSE	r	β
[0.2 11.0]	[0.2 11.0]	0.0020	0.94	0.49
[0.2 11.0]	[0.2 11.0]	0.0035	0.96	0.48
[0.2 5.6]	[5.6 11.0]	0.033	0.0	0.0

Table 2: Second Table .

3.2 Second example

This example has been shown in [6] :

$$g(x_1, x_2) = 1.9(1.35 + e^{x_1} \sin(13(x_1 - .6)^2) \\ e^{-x_2} \sin(7x_2)) + 4. \quad (4)$$

n	card(T)	T	MSE	r	β
49	2500	[0.0 1.0]	0.05	0.26	0.25
4	100	[0.0 1.0]	1.12	1.0	0.26

Table 3: Third Table .

The fuzzy model is pre-validated (β = 0.26, r=1.0) while it gives bad results. It's because the number of rules n is very small(n=4). We see that performances are significantly improved, when n increases.

3.3 Third example

We consider the function :

$$h(x,y) = (1. + x^{(-2.)} + y^{(-1.5)}) * (1. + x^{(-2.)}) + y^{(-1.5)}) \quad (5)$$

n	T	D	MSE	r	β
49	1.0 4.0	1.0 4.0	0.023	0.30	0.66
4	1.0, 4.0	1.0 4.0	2.36	0.30	1.0
4	2.0, 4.0	1.0 2.0	0.7	0.0	0.0
49	2.0, 4.0	1.0 2.0	0.97	0.0	0.0

Table 4: Fourth Table .

We note T = Test set and D = datum plane.

Performances are not increased with n when datum plane and test set are disjoined.

3.4 Fourth example

We consider again the function

$$y(t) = \frac{sin(t)}{t}$$

We have made training (100 iterations) in two cases :

- $\beta = 0.49$
- $\beta = 0.0$

We obtain these results :

β	MSE without tra	MSE with tra
0.49	0.02	0.0005
0.0	0.03	0.03

Table 5: Fifth Table .

We see, in this example that training is not efficient when datum plane is not covered. ($\beta = 0.0$)

4 Conclusion

We have proposed a criterion for pre-validation of a fuzzy inference system. When the model is not pre-validated, we have not to carry out next steps, particularly optimization step.
We have shown that this criterion reflects better that the first one the datum plane's coverage.
From these experiments, we remark that coverage rate is correct under two conditions :

- (i) datum plane and test interval are not disjoined.
- (ii) the ratio number of rules / card(T) is acceptable.

We also note that training is useless when the model is not pre-validated.

References

[1] Remy Viennet. Nouvelle methodologie de planification experimentale pour l'optimisation multicritere de procedes. *These de Doctorat de L'Institut Polytechnique de Lorraine*, 1997.

[2] J. Kleijnen. Statistical theory on design experiments, applied to simulation. *Second International Symposium on Sensitivity Analysis of Model Output,Venice(Italy)*, April 1998.

[3] Tagaki T. Sugeno M. Fuzzy identification of systems and its applications to modeling and control. *IEEE Trans. on SMC*, 15(1), January 1985.

[4] Glorennec P.Y. Algorithmes d'apprentissage pour systemes d'inference floue. *Editions Hermes*, 1999.

[5] J.Gutierrez F.Fernandez S. Guadarrama, E. Trillas. A step towards conceptually improving takagi-sugeno's approximation. *IPMU'2002*, III:1789–1794, July 2002.

[6] Maechler Hwang, Lay. Regression modeling in back-propagation and projection pursuit learning. *IEEE Trans. Neural Networks*, 5(3), May 1994.

A NOVEL NEURO-FUZZY SYSTEM FOR MOBILE ROBOT REACTIVE NAVIGATION

Monaf S. N. Al-Din
Assistant Professor, Department of Mechatronics,
Al-Balqa Applied University, Jordan
E-Mail: msnaldin2001@yahoo.com

keywords: *Neural networks, Fuzzy systems, neuro-fuzzy reasoning, Mobile robot navigation.*

Abstract
This paper presents a novel neuro-fuzzy reasoning system for mobile robot to navigate among static obstacles in unknown environment. The proposed approach has the advantage of greatly reducing the number of if-then rules, that could rise when using classical fuzzy systems, by introducing weighting factors for the sensor inputs, thus inferring the reflexive conclusions from each input to the system rather than putting all the possible states of all the inputs to infer a single conclusion. Four simple neural networks are used to determine the weighting factors. Each neural network is responsible for determining the weighting factor for one sensor input. Simulation results are presented to demonstrate the merits of the proposed system.

Introduction
Among all the soft-computing methods suggested for mobile robot reactive navigation, fuzzy logic systems have been found to be the most attractive. They are tolerant to noise and error in the information coming from the sensory system, and most importantly they are factual reflection of the behavior of human expertise. In general, there are two approaches to the application of fuzzy logic in mobile robot navigation, namely, behavior-based approach [1-2] and classical fuzzy rule-base approach [3-6]. The design of fuzzy logic rules is often reliant on heuristic experience and it lacks systematic methodology, therefore these rules might not be correct and consistent, do not possess a complete domain knowledge, and/or could have a proportion of redundant rules. Furthermore, when a better precision is needed the number of input variables and their fuzzy values need to be increased, e.g., when using four input variables each mapped by seven fuzzy values then 2401 if–then rules are required to define the rule-base of the inference system. Such huge expansion in a multi-dimensional fuzzy rule-based system adds further ad hoc to the design of the system. Several successful reactive navigation approaches based on neural networks have been suggested in the literature [7-8]. In spite of the different suggested network topologies and learning methods, neural reactive navigators perceive their knowledge and skills from demonstrating actions. Therefore, they suffer from a very slow convergence, lack of generalization due to limited patterns to represent complicated environments, and finally information encapsulated within the network can not be interpreted into physical knowledge. Consequently, the utilization of neural networks in reactive mobile robot navigation is limited when compared to fuzzy logic. However, the role of neural networks has been found to be very useful and effective when integrated with fuzzy systems. The birth of this marriage between these two soft-computing paradigms is the neuro-fuzzy systems. Neuro-fuzzy systems provide an urgent synergy can be found between the two paradigms, specifically the capability to mimic human experts as in fuzzy logic, and learning from previous experience capability as in neural networks.
In general, neuro-fuzzy systems can be classified into three categories, neurally adaptive fuzzy inference system, neurally performed FIS, and combinatorial, or hybrid, neuro-fuzzy systems. The first category is the most widely used neuro-fuzzy systems, and they are designed to combine the learning capabilities of neural networks and reasoning properties of fuzzy logic. Further discussion to this issue will be presented in the next coming section.
In this paper a new approach to the design of a simple hybrid neuro-fuzzy navigation system is described. The suggested system has two apparent advantages. First, the if-then rule base is replaced by a set of simple neural networks. Second, inference is on the reflexive conclusions from each input to the system, rather than putting all the possible states of all the inputs to infer a single conclusion. Four parallel simple neural networks are utilized to generate weighting factors for the distance readings acquired by the robot's sensory system. These weighting factors represent the degree of collision avoidance by the robot from a certain side. These weighting factors are then treated as fuzzy values that are input to a defuzzifier to come up with a crisp value for the robot's steering angle and speed.

2. Navigation Methodology
Various algorithms have been proposed to attack the problem of generating collision free trajectories for a mobile robot by utilizing neuro-fuzzy systems [9-14]. One of the methods used in designing a neuro-fuzzy navigation system, is based on training a neural network patterns of sensor readings corresponding to a variety of obstacles [13-14]. Usually in such a methodology complete operator's experience is provided to the network and its training is supervised and performed off-line. Alternatively, partial operator's experience is provided to the network and its training is supervised and performed off-line at the first stage, and then navigator performance is enhanced by using on-line reinforcement learning. It should be noted that a robot may face during its course of navigation a variety of obstacles of different and complicated shapes that could be present in the surrounding, and they could be randomly located and oriented. Hence a huge number of patterns are required for the obstacle recognition methodologies, or a

very long time will be required for unsupervised learning methodologies.

From the above discussion, it is believed that it is better to consider the navigation process to be based on a very simple human experience to generate the reaction of the mobile robot towards the surrounding through neuro-fuzzy reasoning. Such system is based on two facts, first humans cannot get used to all the possible arrangements of obstacles and second cannot build a huge fuzzy model that contains all the possibilities of the 'If...Then...' rules. Instead, they give weighting factors to which direction they are going (left, ahead, right) and what their speed should be, and this possibly can be made based on information from the different senses. The neuro-fuzzy reasoning system suggested in this work, see figure 2, consists of a neural network/s responsible for generating independent certainty weighting factors for the three basic directions (left, ahead, and right) corresponding to instantaneous sensory information. These decisions are then combined, with the same level of simplicity by a diffuzifier, to obtain a final conclusion. The main objective of the proposed method is to reduce the size and time required by a fuzzy inference system by combining the learning capability in neural networks and reasoning capability in fuzzy inference systems, without affecting the efficiency and performance of the navigation system when compared to other classical implementations of reactive fuzzy and neuro-fuzzy navigators.

Learning Methodology and System Structure

The approach is based on divide and concur strategy; where instead of having single multi-input neural network four three-layer neural networks were used. Each network is designed to receive only the distance measured by the corresponding group of ultrasonic sensors from the robot to any possible obstacle that may detected in that direction and generates a weighting factor that represents the degree of certainty to avoid the collision with the obstacles at that side. To generate the required data to train the neural networks a group of operators were required to answer a questionnaire asking them to represent their judgment to the measured distance and the degree of certainty weighting factor in a fuzzy format. Each measured distance was represented by five fuzzy values, Very Far (VF), Far (F), Medium (M), Close (C) and Very Close (VC), while the weighting factors were also represented by five fuzzy values, Very High (VH), High (H), Medium (M), Low (L) and very Low (VL). By averaging all the answers, the universe of discourse of both variables and their representation in terms of fuzzy sets were defined as shown in Figure 1.

The structure of the proposed system is shown in Figure 2. Four input variables are required to provide the necessary information for the navigation system to safely drive the mobile robot to reach the desired target. These inputs are: distances d_f, d_r, and d_l, measured by three ultrasonic sensors. These distances are the distances between the robot and any possible obstacle with respect to the local front, right, and left directions of the robot, respectively. The forth input, d_{vt}, is the distance directed in a global virtual direction between the robot and the target. The outputs of the system are the steering angle θ and the speed of the robot ν. The idea of using a virtual target orientation instead of the real orientation comes from a realistic representation to the behavior of expert driver, where it is impossible for a driver to abandon his attention to the frontal sight when leaving a one-sided blocked target behind him and concentrates on the real target orientation. Under this situation the driver put some concentration towards a virtual orientation at the same side of the target, which should not exceed a certain limit in the range of the frontal sight. Each distance variable from the corresponding sensor is then input to a simple neural network to generate a weighting factor that represents the degree of collision avoidance of the robot from the side of the corresponding sensor. The output weighting factors of each network were values between 0 and 1, with 0 value stating that the robot is very close to an obstacle and a value of 1 stating that the robot is very far from the obstacle. The structure of the four neural networks is identical, where each network consists of a single input node, a single output node, and a single hidden layer with ten nodes. Back propagation has been used to train the networks.

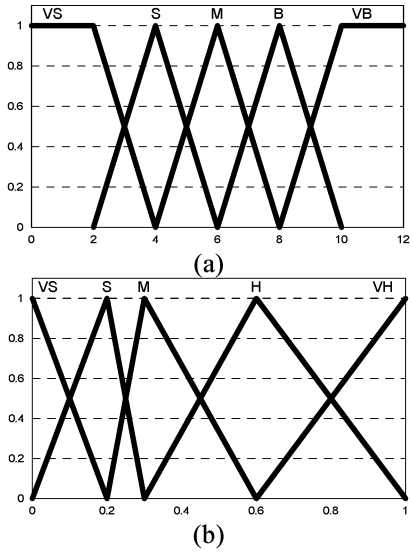


Figure 1: Fuzzy sets representations of (a) sonar measured distance, (b) weighting factor

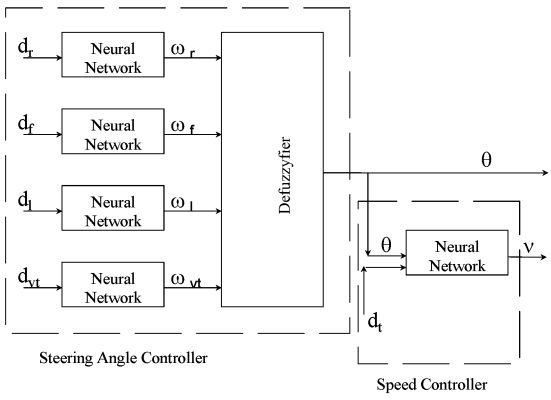
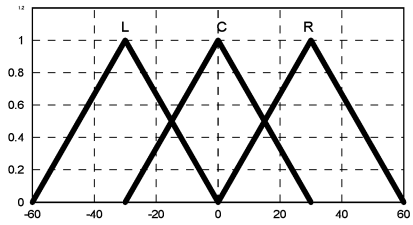


Figure 2: Structure of the proposed system

The middle block in the system is a simple defuzzifier that receives the four weighting factors coming from the previous neural network subsystems, and treats these factors as the degree of fulfillment for the corresponding fuzzy values of the steering angle of the robot. The Center of Area method is used in this block to obtain the final crisp value for the steering angle of the robot. The membership functions for the fuzzy values of the output variable θ are shown in Figure

3. Only three of the fuzzy values are shown in Figure 3, i.e., the turning angle to the left, center, and right, respectively. The fuzzy set that represents the steering angle towards the target orientation is similar except that it is designed to be floating with its center moving in the range [-30°, 30°].



.**Figure 3:** Fuzzy set definition for the output variable θ.

Once the final value for the steering angle θ is obtained, the robot's speed can be computed by a two-input neural network The inputs to the neural network are the steering angle θ obtained from the previous stage and the distance d_t that represents the distance between the robot and the obstacles with respect to a virtual target. The input distance d_t is used to slow down the robot as it approaches the target. The training sets for the neural network were obtained by simulating the robot's motions and estimating the required speeds of the robot for different values of d_t and θ.

Simulation Results

In order to confirm the efficiency of the proposed method, a simulation program with a graphical user interface has been developed. The robot is depicted in the simulation as a circle to resemble a prototype mobile robot that the authors have designed and constructed for experimental purposes. It is noted here, that errors due to wheel slippage and other motion errors were not considered in the simulation. Distance measurements were acquired by four ultrasonic sensors mounted on the robot. Each sensor was modeled by a number of rays within a sector region of a wide beam-angle. The distance measured by each sensor is considered to be equal to the minimum distance obtained within the sector of each sensor while taking into consideration the minimum reliable distance that can be measured by actual ultrasonic sensors.

Four different simulation cases are presented in this section to analyze the reaction behaviors of a mobile robot in avoiding a variety of unknown static obstacles placed randomly in a portion of an unknown environment. The aim here is to study the performance of the proposed approach under the most possible situations. In all these cases the robot is assumed to be initially moving with full speed and its relative steering angle is assumed to be zero. The analysis of the reaction behaviors of the robot is based on observing the instantaneous variation of the four weighting factors and their influence on both the steering angle and speed.

In the First case, Figure 4.a, the robot is passing through an empty L-shaped tunnel where the only present obstacles are the parallel bounding walls of the tunnel. The response of the robot towards the walls is influenced by the instantaneous variation of the weighting factors as shown in Figure 4b-4c. At the first instant the level of cautions towards the obstacles at both sides are equal, while the weighting factors for the front and target sensors indicate that there are no obstacles in either direction, thus the robot will move towards the target.

As the robot proceeds in moving towards the target the steering angle will be gradually reduced because of the continuing increase in the difference between the right and left weighting factors and the fall of both front and target weighting factors. Once the robot becomes close to the right wall the left weighting factor will rebalance the congregated right and target weighting factors. Thus, the robot will slightly turn to the left until it aligns itself to move later in parallel with the right wall. When reaching the end of the tunnel, the target weighting factor will rapidly increase to 1. Hence, the robot will noticeably reduce its speed for a short while until it is completely turned in the direction of the target.

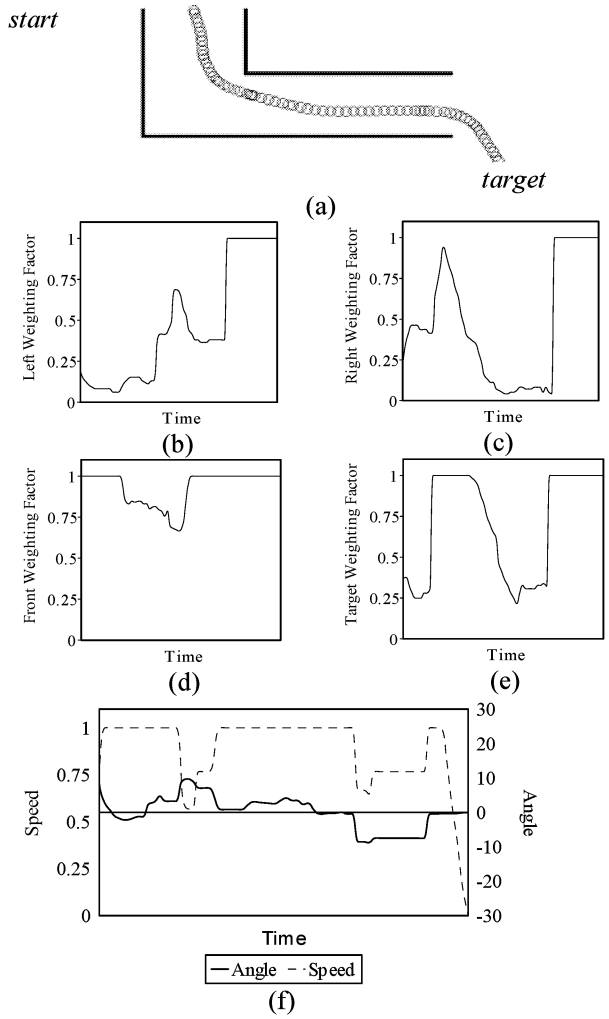


Figure 4: Behavior of weighting factors, speed and steering angle while the robot is moving through an L-shaped tunnel.

The second case, Figure 5, the robot is heading towards the target at full speed when a square obstacle blocks its path. The obstacle totally blocks the robot from the right side and slightly extended above the line that connects the location of the robot and the target. As it can be seen in Figure 5 the first apprehension is from both the front and target sensors through the decreasing values of their corresponding weighting factors. The right sensor detects the presence of the obstacle, and the robot immediately reacts by turning gradually to the left while reducing its speed due to the dominance of the left weighting factor (see Figure 5.f). Once the robot passes the obstacle both the front and the target

weighting factors increase sharply. The right weighting factor follows and rises sharply to indicate the absence of any obstacle in all directions. Reacting immediately to this situation, the robot reduces its speed and turns to the right side to align itself again with the target direction.

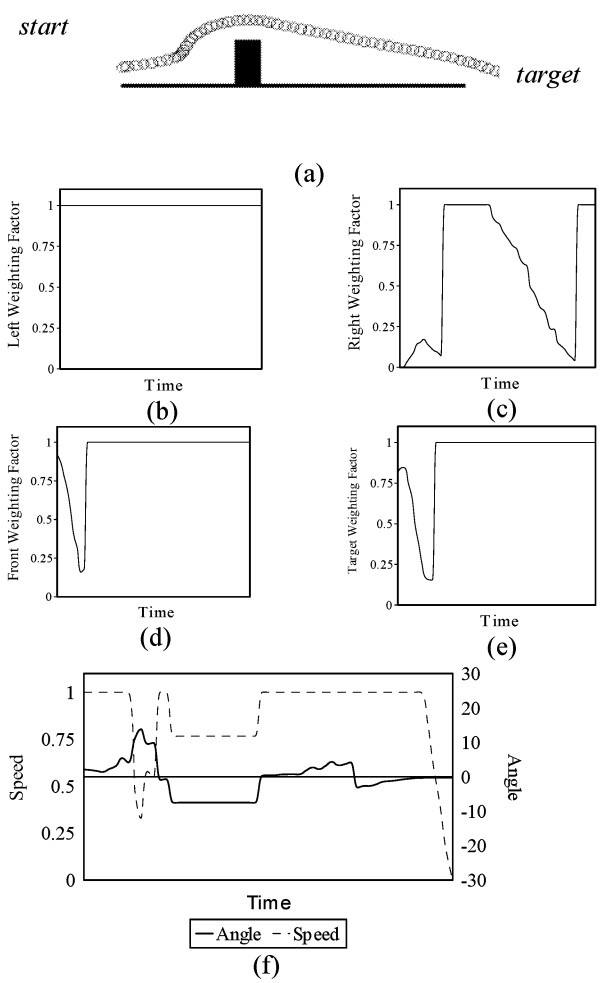


Figure 5: Behavior of weighting factors, speed and steering angle while the robot is blocked by a simple obstacle.

The next case presented is depicted in Figure 6. The case describes the situation when the robot is moving in a room while the target is in another room. The behavior of the robot in the beginning is similar to that in the previous case until the robot is faced with the corner walls. At this time only the right weighting factor is active will all other factors are zero. Consequently, the robot turns to the right. Once the robot turns to the right its path becomes blocked from both the left and front again. Hence, it keeps on turning to the right until the front weighting factor is active. The robot keeps on moving in the same direction until the right weighting factor is 1 at which time the robot turns back towards the target. The variation of weighting factors, speed and steering angle are shown in Figure 6.

The final case, Figure 7, presents the situation where the robot is trapped by a dead-end. The behavior of the robot in this case is very much similar to that of the previous case. When faced by the dead-end, the robot starts turning left under the effect of the assisting rule. Once the robot turns left, the other side of the concave obstacle will block its path. Hence the robot keeps on turning left until it is totally away from the target. Furthermore, due to the narrowness of the tunnel, the robot will keep on moving away from target until it gets close to the opening. At this time the target attraction behavior becomes dominant and the robot turns and moves until it reaches the target. The variation of weighting factors, speed and steering angle are shown in Figure 8.

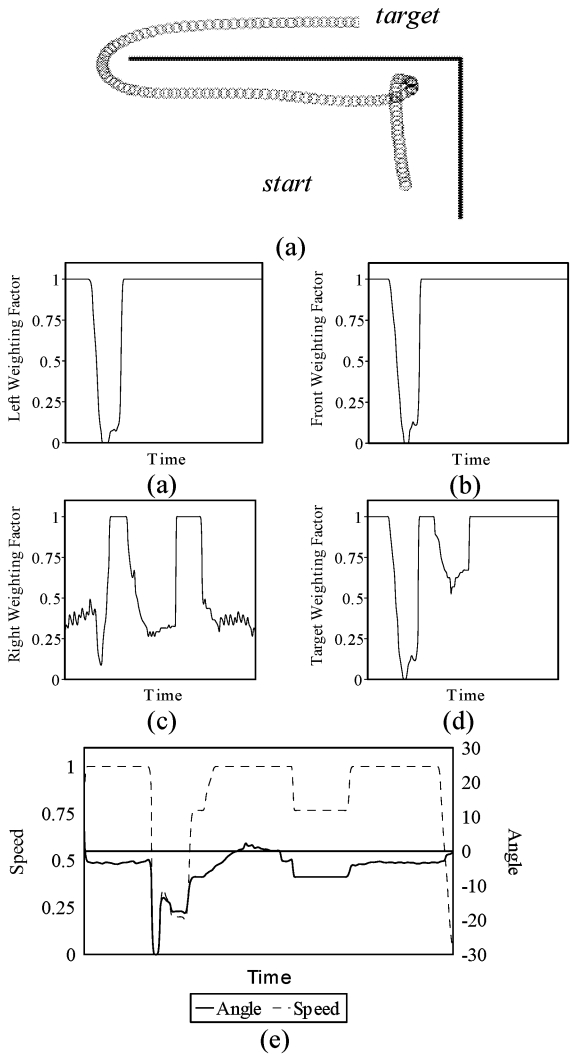


Figure 6: Behavior of weighting factors, speed and steering angle while the robot is blocked by a long obstacle.

Finally, Figures 8 and 9 present more complicated simulation case to show the performance of the proposed reactive neuro- fuzzy navigator system towards more realistic situations.

Concluding Remarks

A neuro-fuzzy reasoning scheme for mobile robot navigation has been presented in this work. The approach is based on decomposing a multidimensional fuzzy system into a set of simple parallel neural networks. This method relies upon finding quantifiable means to represent the expert's experience, and determines a mapping from current state of a system to the fuzzy measures with which the expert's knowledge is quantified. The concept of weighting factors for the sensor inputs expressing the reflexive conclusions of each input rather than having to go through a huge list of rules to infer a single conclusion is introduced here for the first time. Therefore, the method has the advantage of replacing the huge number of "If-Then" rules by simple

parallel neural networks. The approach was tested in a number of simulated case problems to demonstrate its effectiveness, and it was found that the results compromise with reasonable satisfaction the obstacle avoidance and target reaching requirements. In addition to that, the proposed controller showed the capability of a mobile robot to escape from simple traps.

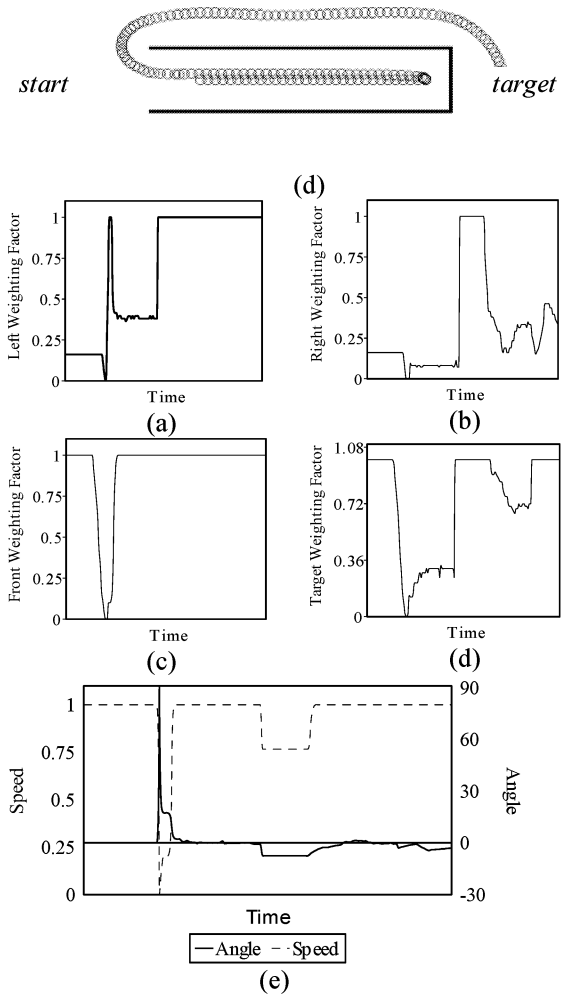


Figure 7: Behavior of weighting factors, speed and steering angle while the robot is trapped by a dead-end

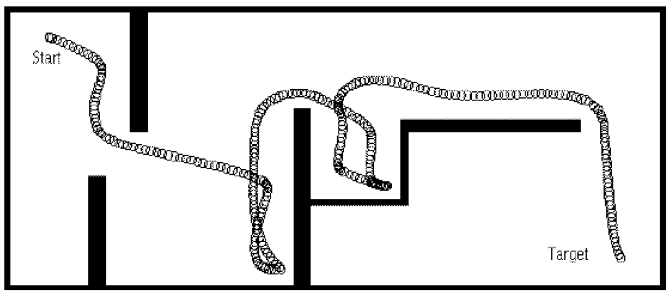


Figure 8: Simulated Trajectory of a MR in a maze (1)

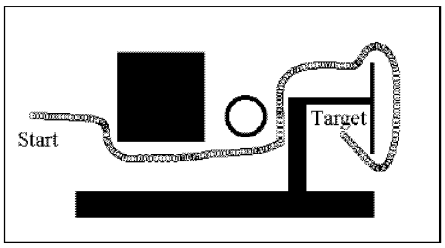


Figure 9: Simulated Trajectory of a MR in a maze (2)

References

1. A. Saffiotti, E.H. Ruspini, and K. Konolige, "Using Fuzzy Logic for Mobile Robot Control", In Practical Applications of Fuzzy Technologies, Edt. by H.J. Zimmermann, Kluwer Academic, 1999, pp. 185-206.
2. E. Aguirre and A. Gonzalez, "Fuzzy Behaviors for Mobile Robot Navigation: Design, Coordination and Fusion", International Journal of Approximate Reasoning, Vol. 25, Issue 3, 2000, pp. 255-289.
3. R. Chatterjee and F. Matsuno, "Use of Single Side Reflex for Autonomous Navigation of Mobile Robots in Unknown Environments", Robotics and Autonomous Systems 35, 2001, pp. 77-96.
4. P.G. Zavlangas, S.G. Tzafestas and K. Althoefer, "Fuzzy Obstacle Avoidance and navigation for Omni directional Mobile Robots", ESIT 2000, Aachen, Germany, 2000, pp. 375-382.
5. W.L. Xu, S.K. Tso and Y.H. Fung, "Fuzzy Reactive Control of a Mobile Robot Incorporating a Real/Virtual Target Switching Strategy", ibid. 23, 1998, pp.171-186.
6. W.L. Xu, "A Virtual Target Approach for Resolving the Limit Cycle Problem in Navigation of a Fuzzy Behavior-Based Mobile Robot", ibid. 30 , 2000, pp. 315-324.
7. Nadine Tschichold-Gurman, "The Neural Network Model RuleNet and its application to Mobile Robot Navigation", Fuzzy Sets and Systems 85, 1997, pp. 287-303.
8. Dhiraj Bajaj, and Marcelo H. Ang Jr., "An Incremental Approach in Evolving Robot Behavior", Proc. of Sixth Int. Conf. on Control, Automation, Robotics and Vision, (ICARCV 2000), 2000.
9. Takatoshi Nishina and Masafumi Hagiwara, "Fuzzy Inference Neural Network", Neurocomputing 14, 1997, pp. 223-239.
10. Jelena Godjevac and Nigel Steele, "Neuro-Fuzzy Control of a Mobile Robot", Neurocomputing 28, 1999, pp. 127-143.
11. G. N. Marichal, L. Acosta, L. Moreno, J.A. Mendez, J.J. Rodrigo, and M. Sigue, " Obstacle Avoidance for a Mobile Robot: A Neuro-Fuzzy Approach", Fuzzy Sets and Systems 124, 2001, 171-179.
12. Xiaowei Ma, Xiaoli Li, and Hong Qiao, "Fuzzy Neural Network-Based Real-Time Self-Reaction of Mobile Robot in Unknown Environments", Mechatronics 11, 2001, pp. 1039-1052.
13. Wei Li, Chenyu Ma and F.M. Wahl, "A Neuro-Fuzzy System Architecture for Behavior-Based Control of a Mobile Robot in Unknown Environments", Fuzzy Sets and Systems 87, 1997, 133-140.
14. Kai-Tai Song and Liang-Hwang Sheen, "Heuristic Fuzzy-Neuro Network and its Application to Reactive Navigation of a Mobile Robot", ibid 110, 2000, 331-340.

TIME SERIES CLASSIFICATION BASED ON DISCRETE-SPACE FEATURE EXTRACTION

Željko Jagnjić
Faculty of Electrical Engineering
University of Osijek
k.Trpimira 2b
HR-31000 Osijek, Croatia
E-mail: jagnjic@etfos.hr

Nikola Bogunović
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3
HR-10000 Zagreb, Croatia
E-mail: nikola.bogunovic@fer.hr

Franjo Jović
Faculty of Electrical Engineering
University of Osijek
k.Trpimira 2b
HR-31000 Osijek, Croatia
E-mail: jovic@etfos.hr

KEYWORDS
Time-series analysis, Qualitative, Decision, Parameter identification, Model design

ABSTRACT

In this paper we present a time-series classification method based on discrete-space feature extraction. The main characteristics of the method are expansion and coding of quantitative time-series data. Expansion and coding will result in creation of qualitative difference vector. Qualitative difference vector conveys the full information on the variation of the particular time-series and can be seen as a single point in n-dimensional discrete-space. From discrete-space, symbolic and numeric features are extracted and used for the decision tree construction that is later used in time-series classification. The proposed method was tested in the context of Control Chart Pattern data, which are time-series used in Statistical Process Control. Obtained results are compared with other methods.

INTRODUCTION

Time-series are a form of data occurring in virtually every scientific discipline. Important time-series include stock market prices, sales of a product, all kinds of scientific results, weather readings, medical records and so on. When dealing with time-series data there are two main objectives: prediction of future behaviour based on past behaviours and description of time-series data. Description of time-series data can be used for generalization, clustering and classification. Time-series classification involves learning of a function that maps a series into a class from a set of predefined classes. It is necessary to define a set of well-defined classes. Time-series classification finds application in analysing medical records (electrocardiogram data), different weather readings (storm-cloud data), process monitoring (sensors reading) and statistical process control. A common task among all mentioned categories is the problem of feature extraction and similarity measure calculation. There are numerous different methods in literature covering those two problems. Agrawal, (Agrawal et. al., 1995a;), Singh (Singh, 1998) and Yi (Yi, 2000) represent time-series by few coefficients of their Discrete Fourier Transform. Other approaches use statistical features for describing time-series data (Koegh & Smyth 1997; Sebastian et al, 1999, Alcock & Manolopoulos, 1999). Recent works on feature extraction include time-series shape description using different alphabets (Guralnik, 1997; Singh, 1998) or primitive shapes (Das, 1998). The most important work in this group is definition of Shape definition language (Agrawal et. al., 1995b) and the introduction of qualitative similarity index method by Cuberos (Cuberos at all, 2002).

In time-series literature, the most common technique for evaluating similarity between two sequences is to calculate Lp distance between two sequences $\vec{X}$ and $\vec{Y}$, both of length n. Elements of the sequences can be either real (actual) data or extracted features. Lp distance is defined as:

$$L_p(\vec{x},\vec{y}) = \sum_{i=1}^{n} | x_i - y_i |^p \qquad (1)$$

For p=1 we have Manhattan distance while for p=2 we have Euclidian distance. Given this definition of similarity, most research in the area has concentrated upon finding similar sequences as quickly as possible. Usual problem that arises when using Euclidian distance is the problem of different amplitude scaling and/or time shifting. There are many examples where two similar sequences are not close using the Euclidian distance metrics. To overcome the shortcomings of using Euclidian distance different transformations have been proposed. Such transformations include shifts (vertical and horizontal), scaling (linear and amplitude) and removal of non-matching parts. Problem is that number of transformations that a sequence could undergo is infinite and searching for the correct sequence of transformations would become to long.

Humans, on the other side, prefer reasoning and classification on a more abstract, symbolic level. We believe that qualitative information embedded in the graphical form of tabular time-series data is a key for proper similarity calculations. We propose the creation of qualitative discrete-space based on time-series data expansion and coding. From such defined discrete-space symbolic and numeric features are extracted and used for decision tree construction that is later used for time-series classification.

In the following section we give a brief description of the discrete-space construction. We continue by describing feature extraction from discrete-space. At the end we give some experimental results related to some other methods. The last section concludes the paper and gives some directions for the future work.

DISCRETE-SPACE CONSTRUCTION

Discrete-space construction is the main feature of the proposed method. It consists of three steps: (i) time-series data expansion, (ii) coding and (iii) qualitative matrix construction. In the following paragraphs each step will be briefly explained.

A time-series is a sequence of real values, each one representing the value of a magnitude at a point of time. Time-series will be denoted as $\vec{X}=(x_1, x_2, ..., x_n)$. For time-series with n members we will say that time-series length is n. Since we are interested in qualitative relation among time-series members appropriate transition from quantitative domain into qualitative domain should be done. Method called expansion of input data stream by evolution of input data difference vector to the right is used. The method was already used in qualitative-quantitative process modelling (Jagnjić, 2001, Bogunović at. all., 2002). If we denote by Δ_{ij} sequential difference between *i* and *j* member of observed time-series $\vec{X}$

$$\Delta_{ij}=x_i-x_j, \ \forall \ i,j; \ i<j \qquad (2)$$

then the difference vector $\vec{\Delta}$ conveys the full information on the variation of the particular time-series. The cardinality of a new vector amounts to

$$m=\frac{n(n-1)}{2}. \qquad (3)$$

Difference vector $\vec{\Delta}$ will be: $\vec{\Delta}=[\Delta_{12}, \Delta_{13}, ..., \Delta_{1n}, \Delta_{23}, \Delta_{24}, ..., \Delta_{(n-1)n}]$.

Let us assume a short time-series $\vec{X}$ with five members: $\vec{X}=(2, 4, 3, 7, 4)$. According to the equation (3) cardinality of difference vector $\vec{\Delta}$ will be 10. The complete difference vector $\vec{\Delta}$ will be: $\vec{\Delta}=[-2, -1, -5, -2, 1, -3, 0, -4, -1, 3]$.

After expansion, the coding step is performed. It can be seen that elements of the difference vector $\vec{\Delta}$ are positive, negative and zero numbers. Since we are interested only in qualitative information it is natural to employ the following coding:

$$\Delta_{ij}>0 \rightarrow \text{'10'} \qquad \Delta_{ij}<0 \rightarrow \text{'01'} \qquad \Delta_{ij}=0 \rightarrow \text{'00'}$$

Applying the proposed coding to difference vector $\vec{\Delta}$, new qualitative difference vector $\vec{\Delta}^C$ will be created:

$\vec{\Delta}^C=[01, 01, 01, 01, 10, 01, 00, 01, 01, 10]$

Qualitative difference vector $\vec{\Delta}^C$ contains complete qualitative information about relation between any two members of the observed time-series $\vec{X}$. Also, qualitative difference vector $\vec{\Delta}^C$ can be viewed as m-dimensional discrete space, where each time-series of length **n** will occupy a single point.

As the proposed vector representation could cause some misunderstandings, it is not appropriate for future analysis. Consequently, a simple transformation into a qualitative matrix is done. From the expansion equation (2), complete qualitative difference vector $\vec{\Delta}^C$ can be observed through the introduction of *section.* Each section will represent relation between one member and all the other successive members from the observed time-series. For time-series with length **n**, the number of sections will be:

$$NS = \mathbf{n} - 1 \qquad (4)$$

The first section will have (n-1) elements and will represent qualitative relation between the first member of the observed time-series and all other successive members. The second section will have (n-2) elements and so on till the last section with only one element. Qualitative difference vector $\vec{\Delta}^C$ from our example will be divided into the following four sections.

$\vec{\Delta}^C=[01, 01, 01, 01 \mid 10, 01, 00 \mid 01, 01 \mid 10]$

1st 2nd 3rd 4th section

Each section can be put in one matrix row. The final result is matrix whose dimension will be NS*NS. Such defined matrix will be called qualitative matrix. Qualitative matrix for our example will be:

QM=

01	01	01	01
	10	01	00
		01	01
			10

Representation of the qualitative difference vector $\vec{\Delta}^C$ by the qualitative matrix form provides the means for a new approach to time-series analysing. The matrix can be analysed in different ways. For example, elements along the main diagonal represent series of differences usually obtained through the standard procedure when only differences between two neighbours are calculated. Next, if we remove set of columns, we actually broke time-series at some point. Removing the last column, we observe time-series with the first four members only. The same is with rows. Removing rows actually means that we are interested in members that follow. Also we can easily separate time-series members between any two points. If we remove the first row and the last column, data will represent situation between the second and the fourth member and so on. Also, construction of the qualitative matrix sets up abundant possibilities for numeric and symbolic features extraction.

FEATURE EXTRACTION

We have already mentioned that humans prefer reasoning and classification on a more abstract, symbolic level. With the construction of qualitative matrix the first step into that direction is done. Any two members from time-series are analysed through their relative relation only. Any member can be greater than, smaller than or equal to some other member. Moreover, humans prefer visualization rather than tabular representation. For the purpose of visualization, coding elements inside the qualitative matrix can be represented with colours. Visualization process of qualitative matrix helped us in establishing feature set that we later used for decision-tree construction and time-series classification. Since we applied coding with three different states, three main colours were used: red, blue and green. Red colour replaced code element '10', blue colour replaced code element '01' and green colour replaced code element '00'. The coloured qualitative matrix for our example is shown on Figure 1.

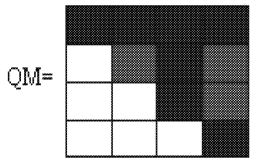


Figure 1: Coloured Qualitative Matrix

The qualitative matrix can be observed as bitmap where each coded element represents one pixel. For the creation of the base feature set, human subjects were interogated using six different images. Those images were created using our approach applied to six different time-series identified as Control Chart Pattern data and used in previous work by Alcock & Manolopoulos (Alcock & Manolopoulos, 1999), Pam & Chan (Pam & Chan, 1998) and Nanopoulos (Nanopoulos at all. 2001). Test images are shown on Figure 2.

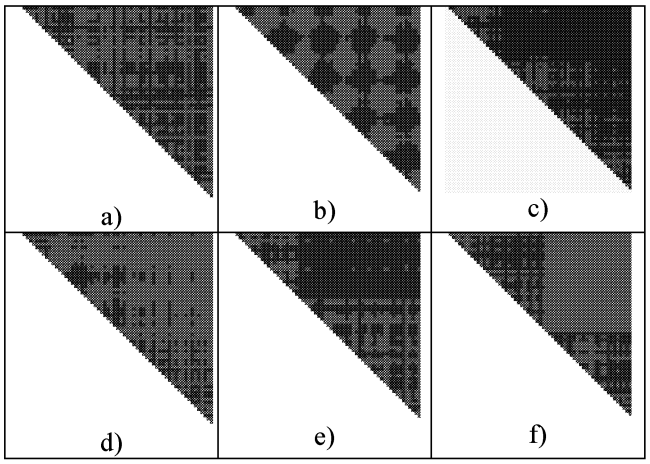


Figure 2: Images used for Human Testing

The original time-series were of length 60. Complete description of each time-series will be given in the next section. About 100 students were tested. During the test they were encouraged to describe images, one by one, and at the end all six images were presented. Although there were small percent of poetic descriptions like see, clouds, computer chips, town from the bird perspective, etc., most of them described images in form of different features. The most recurrent features extracted from images are presented in Table 1.

It can be seen from Table 1 that each image has some unique features that distinguish it from other images. Also it is very interesting to note the term "inverted". Since student saw only two colors in images, red and blue, they become inverted colors. Based on features from table 1, a complete set of basic features was proposed for decision tree construction and time-series classification. It consists of 31 attributes and class definition depicted in Table 2.

We will not explain all features in detail. In general, feature can have some numeric value or can have some nominal value. For example, the existence of dominant segment is described in terms of boolean values True or False. Also, the position of dominant segment is described in terms of N(Unknown), A(left upper corner) or B(right lower corner). A dominant image color describes general trend of time-series. If red is dominant color than the observed time-series trend can be identified as decreasing.

Table 1: Features used for Image Description

Image: a	Equal quantity of red and blue colour.
	Different shapes
Image: b	Equal quantity of red and blue colour.
	Blue and red cyclic (elliptic) shapes
Image: c	Blue is dominant colour
	Different red unknown shapes (small size)
Image: d	Red colour is dominant
	Different blue unknown shapes (small size)
	Same image as image c with inverted colours
Image: e	Dominant blue square in upper right corner of image
	Two triangles in image corners similar to image a
Image: f	Dominant red square in upper right corner of image
	Two triangles in image corners similar to image a
	Same image as image e with inverted colours in square

Table 2: Complete Basic Feature Set

General attributes	Number of red pixels {real value} Number of blue pixels {real value} Number of green pixels {real value} Dominant image colour {red, blue, green, unknown}
Attributes for (red, blue, or green) pixels	Number of segments {real value} Dominant segment {True, False} Dominant segment area {real value} Dominant segment position {N,A,B} Group of segments {True, False} Numb. of seg. in group {real value} Average area of group {real value} Numb. of other seg. {real value} Average area of other seg. {real value}
Class	{Normal, Cyclic, Increasing, Decreasing, Upward, Downward}

For the generation of some features (attributes), different image filters and eroding algorithms were used. We will not explain them here. Details can be found in Costa & Cesar (Costa & Cesar, 2001). Due to image simplicity and only three colors (three different direction of change) those algorithms were very simple. The proposed basic feature set was later used for decision tree learning and time-series classification.

EXPERIMENTAL EVALUATION

The proposed method was tested in the context of Control Chart Pattern data, which are time-series used in Statistical Process Control. Control Chart Pattern data are used because classification can be performed very effectively since they contain well-described classes. Beside that, Control Chart Pattern data possess quantifiable similarity. The Control Chart Patterns described in Pan & Chan (Pan & Chan, 1998) and Nanopoulos (Nanopoulos, 2001), are artificially generated by six equations. Each equation represents a different type of pattern. The six pattern types are shown in Figure 3 and are described as normal, cyclic, increasing trend, decreasing trend, upward shift and downward shift.

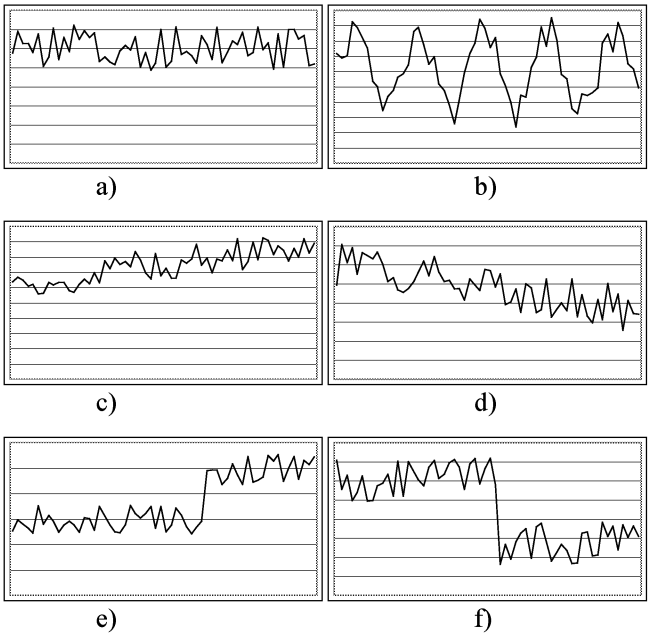


Figure 3: Pattern Types

Letters a), b), c), d), e) and f) correspond with the same letters as in Figure 2. The similarity between images given in Figure 3 and Figure 2 is obvious. For the experiments, 600 patterns were used, 100 of each type with length of 60. These patterns were downloaded from UCI KDD archive (Hettich and Bay, 1999). For the feature generation a new computer program solution was written in C/C++. For decision tree construction and time-series classification system called Weka (Waikato Environment for Knowledge Analysis) was chosen (Witten & Frank, 2000). Weka is freely available on the World Wide Web and has integrated numerous types of classifiers. We used the J4.8 algorithm, which is Weka's implementation of C4.5 decision tree learner. The results of our method were compared with three other distinctive classification methods: classification based on typical Euclidian distance calculation, classification based on feature-based Euclidian distance calculation (Alcock & Manolopoulos, 1999) and classification with Multi-layer Perceptron neural network (Pham & Oztmel, 1992). The employed performance measure was:

$$PM = \frac{T - F}{T + F} \quad (5)$$

where T is the number of patterns found to be similar which are actually similar (true positives) and F is the number of patterns incorrectly found to be similar (false positives). The ideal performance measure is obviously PM = 1.

Our method was tested using two different approaches. In the first approach no test file was given, so stratified ten-fold cross validation was performed. Confusion matrix for the first approach is presented in Table 3.

Table 3: Confusion Matrix

a	b	c	d	e	f	<-- classified as
100	0	0	0	0	0	a = Normal
0	99	0	0	1	0	b = Cyclic
0	0	95	0	5	0	c = Increasing
0	0	0	98	0	2	d = Decreasing
2	0	5	0	92	1	e = Upward
0	1	0	7	0	92	f = Downward

Each matrix element shows the number of test examples for which the *actual* class is the row and the *predicted* class is the column. Good results correspond to large numbers down the main diagonal and small, ideally zero, off-diagonal elements, which is actually presented in table 3.

In the second approach training and test set were generated. The training set consisted of 66 patterns from each of the six pattern types. The test set consisted of 34 patterns from each of the six types. Patterns in training and test sets were randomly chosen and complete procedure was run ten times and the average performance was taken. Confusion matrix for the second approach was very similar to the first approach and it will not be presented here. The obtained results for all classification methods are presented in Table 4.

Table 4: Experiment Results

Method	**PM**	**Accuracy**
Euclidian distance	0.371	65%
Feature-based Euclidian distance	0.473	73%
Multi-layer perceptron	0.918	96%
Discrete-space feature based (approach a)	0.916	95.8%
Discrete-space feature based (approach b)	0.893	94.6%

As it can be seen from Table 4, the proposed method exhibited significantly better results than classification methods based on pure Euclidian distance calculation. The results obtained with our method can be compared with results obtained with multi-layer perceptron neural network. Taking into consideration usual problems when dealing with neural networks (different parameters estimation and time necessary for learning, no obvious correlative knowledge generation) the proposed discrete-space feature extraction method outperforms all the other mentioned methods.

CONCLUSION

In this paper we have proposed a discrete-space feature extraction method for time-series classification. The method is based on quantitative data expansion and coding. The result of those two steps is qualitative difference vector, which is later transformed into the qualitative matrix. The qualitative difference vector can be observed as a single point in m-dimensional discrete-space. The construction of qualitative matrix enables time-series data analysis on more abstract and symbolic level. From the qualitative matrix, various numeric and symbolic features are extracted and later used for decision tree construction and time-series classification. The basic set of features was generated with the help of human subjects through the test procedure. For better understanding and visualization, the qualitative matrix can be easily colored with only three different colors. The method was tested in the context of Control Chart Pattern data, which are time-series used in Statistical Process Control. The observed time-series can be classified as short (only 60 members each). However, they maintain significant similarities and the usual classification method based on pure numeric calculation can't perform well. The results obtained with our method are promising and can be compared with multi-layer perceptron neural network.

Future work could be carried out in many directions. First of all, the proposed method should be tested on much longer and real data. From our point of view it can be expected that such time-series will demand some new features to be added to the basic set. Also, the possibilities of subsequence matching should be further investigated. The qualitative matrix provides a good foundation for the time-series analysis domain.

REFERENCES

Agrawal, R., Lin, K.I., Sawhney, H.S., Shim, K., 1995., "Fast similarity search in the presence of noise, scaling, and translation in time series databases", *The 21st VLDB Conference*, Switzerland, pp.490-510

Agrawal, R., Psaila, G., Wimmers, E.L., Zait, M., 1995., "Querying shapes of Histories", IBM Research Report RJ 9962 (87921), IBM Almaden Research Centre, San Jose, California

Alcock, R.J., Manolopoulos, Y., 1999., "Time-Series Similarity Queries Employing a Feature-Based Approach", In *Proceedings of 7th National Computer Conference*, Ioannina, Grece, pp.III.1-9.

Bogunović, N., Jagnjić, Ž., Jović, F., 2002., "Qualitative Modeling of Complex System based on Similyrity Calculus", In Amborski, K., Meuth, H., Eds., *Proceedings of the 16th European Simulation Multiconference 2002*, ESM2002, Darmstadt, Germany, June 3-5, 2002, pp. 312-316, ISBN: 90-77039-07-4

Costa, L.F., Cesar, R.M., 2001., *Shape Analysis and Classification: Theory and Practice*, CRC Press LLC

Cuberos, F.J., Ortega, J.A., Gasca, R.M., Toro, M., 2002., "QSI – Qualitative Similarity Index", In *Proceeding of 16th International Workshop on Qualitative Reasoning QR2002*, June 10-12, Spain

Das, G., Lin, K.I., Mannila, H., Renganathan, G., Smyth, P., 1998., "Rule Discovery from Time-Series", In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 16–22.

Guralnik, V., Wijesekera, D., Srivastava, J., 1998., "Pattern Directed Mining of Sequence Data", In *Proceedings of the Fourth International Conference on Knowledge Discovery in Databases*, pp. 51–57.

Hettich, S. and Bay, S. D., 1999., The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science

Jagnjić, Ž., 2001., "Qualitative-quantitatve process modeling by expansion and coding method", *Master thesis*, Faculty of Electrical Enfineering and Computing, University of Zagreb

Koegh, E., Smyth, P., 1997., "A probabilistic approach to fast pattern matching in time series databases", In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, pp.24-40, AAAI Press

Nanopoulos, A., Alcock, R., Manolopoulos, Y., 2001., "Feature-based Classification of Time-series Data", *International Journal of Computer Research*, Nova Science Publishers, pp.81-108

Pham, D.T., Chan, A.B., 1998., "Control Chart Pattern Recognition Using a New Type of Self-Organising Neural Netowork", In *Proc. Instn. Mech. Engns.*, Vol.212, No.1, pp.115-127

Pham, D.T., Oztemel, E., 1992., "Control Chart Pattern Recognition Using Neural Networks", *Journal of System Engineering*, Vol.2, pp.256-262

Sebastian, P., Ramoni, M., Cohen, P.R., Warwick, J, Davis, J., 1999., "Discovering dynamics using bayesian clustering", In *Proceedings of the 3rd International Symposium on Intelligent Data Analysis*, Amsterdam, pp. 199-209

Singh, S., McAtackney, P., 1998., "Dynamic Time-Series Forecasting Using Local Approximation", In *Proceedings of the IEEE Tenth International Conference on Tools with Artificial Intelligence*, pp. 392–399

Witten, I.H., Frank, E., 2000., *Data Mining*, Morgan Kaufmann Publishers, Academic Press

Yi, B.K., Sidiropoulos, N.D., Johnson, T., Biliris, A., Jagadish, H.V., Faloutsos, C., 2000., "Online Data Mining for Co-Evolving Time Sequences", In *Proceedings of the IEEE Sixteenth International Conference on Data Engineering*, pp. 13–22

ŽELJKO JAGNJIĆ was born in Osijek, Croatia and went to University of Zagreb, where he studied electrical engineering and computing. He obtained B.Sc.E.E. and M.Sc.C.S. form the same university in 1997 and 2001 respectively. From 1998 he is working at Faculty of Electrical Engineering, University of Osijek, at Department for Computing at Laboratory for Artificial Intelligence as young researcher. His research interests encompass qualitative methods of modelling and development of fast algorithms for data analysis based on artificial intelligence methods and procedures for reasoning about complex systems. He is also interested in development of intelligence engines for games.

NEURO FUZZY MODELLING WITH EXPERT SYSTEMS

SIMULATION AND MODELLING OF CONCURRENCY CONTROL ENHANCEMENT THROUGH AN ADAPTIVE NEURO-FUZZY INFERENCE EXPERT SYSTEM

Munib Qutaishat
University of Jordan
KASIT, Department of Computer Information Systems
Amman, Jordan
E-mail: maq@ju.edu.jo
munib_qutaishat@yahoo.com

KEYWORDS

Database management systems, Expert systems, Simulation, Fuzzy systems, Neural network

ABSTRACT

The research work reported here is primarily prompted by the need to avoid conflicting transactions and provide a system that enhances concurrency control mechanisms. The performance of transaction processing systems with traditional two-phase locking (2PL) can be degraded by transaction blocking. This may result from the high volume of lock conflicts and aborts used to resolve conflicts.

To this end, an Adaptive Neuro-Fuzzy Inference approach is herein investigated, in which frequent conflicting transactions can be eliminated. For this purpose we have developed a simulation model called Adaptive Neuro-Fuzzy Conflict Resolution Inference System (ANF-CRIS). The proposed system forms a class of adaptive networks that are functionally equivalent to fuzzy inference expert system. The system behavior was tested against a previously developed approach named "Dynamic Attributes". The "Dynamic Attributes" approach depends on the selection of a set of attributes and the priorities of Dynamic Attributes of transactions in solving conflicts. Our system avoids favoritism, as it allows each attribute to participate in the decision of 'how' the conflict should be resolved.

INTRODUCTION

Concurrent access to data is a major issue for databases, especially when a database performs a large number of transactions in a multi-user environment. Concurrency control mechanisms are needed to ensure non-interference or isolation of concurrently executing transactions (Elmasri and Navathe, 2000; Lewis *et. al.*, 2002; Silberschatz *et. al.*, 2002) .

Locking methods are the most widely used approach in handling concurrency control in Database Management Systems (DBMSs) (Bernstein and Newcomer, 1997; Gray and Reuter, 1993; Kumar and Hsu, 1998; Kung and Robinson, 1981). There are several variations of locking methods, which all share common feature which is the way locking protocol is utilized (i.e. read/write lock on a data item prior to the execution of the corresponding read or write operation on that data item).

The two-phase locking protocol was introduced by Eswaran *et. al.* (1976). A large number of concurrency control mechanisms based on a two-phase locking protocol (2PL) have been developed in the past and their performances have been reported in literature (Bernstein and Newcomer, 1997; Bernstein *et. al.*, 1987; Gray and Reuter, 1993; Harder and Rothermel, 1993; Kung and Robinson, 1981; Lewis *et. al.*, 2002; Reed, 1983). The strict two-phase locking protocol is widely used in commercial products to ensure serializability. The limited concurrency offered by traditional strict 2PL protocol becomes a major restriction for applications to deliver high performance. To address this issue, some researchers have proposed to place more structures and semantics on concurrent algorithms (Kuo and Mok, 2000; Mak and Wong, 2000). The main advantage of using semantic information is that the conflict probability can be reduced. Other approaches utilize roll-back and/or blocking mechanism(s) to resolve conflicts and deadlocks for serializing concurrent executions.

Most of the above conflict resolution approaches are both not easily augmented with suitable semantics for conflicts resolution process and incapable of identifying the 'suitable' or 'right' transaction for applying roll-back or blocking (Kuo and Mok, 2000; Mak and Wong, 2000). A suitable transaction and possibly augmented with additional semantics can be regarded as the one which generates a minimum roll-back and/or blocking cost. Therefore, the cost of conflict resolution can be minimized if the suitable transaction is selected and the

minimum cost activity is applied. Otherwise, the performance of transaction processing systems with traditional conflict resolution mechanisms can be degraded by transactions blocking due to the potentially high volume of lock conflicts and aborts to resolve these conflicts.

Dealing with such a situation cannot be determined in advance due to its fuzziness because transaction abort and roll-back depend on many factors including the selection of right transactions and on the run-time progress of transactions. In an attempt to solve this problem a system model must be constructed. System modeling based on conventional mathematical tools (e.g. programmed parameters and differential equations) are not well suited for dealing with ill-defined and uncertain systems (Cornelius, 1998; Ermine, 1995; Kacprzyk, 1997; Haykin, 1999; Pedrycz, 1996; Russo and Jain, 2001, Zadeh, 1965). By contrast, a fuzzy inference system employing fuzzy if-then rules can model the qualitative aspects of human knowledge and reasoning process. In addition, this can be combined with an effective method for tuning the membership functions (MF) to enhance quantitative analysis, minimize the output error measure, and maximize performance.

ANF-CRIS DESIGN GOALS

Concurrency Control Design Goals

The design of concurrency control mechanisms is a difficult task, as it requires both general knowledge about database systems and transaction design, and specific knowledge about a particular conflict-resolution approach (Chen, 1995; Lee and Lam, 2000; Lewis *et. al.*, 2002; Palamidessi, 2000). The former type of knowledge requires general knowledge relevant to the design of transactions, for instance, properties of a transaction and various useful techniques that help in the design of good quality concurrency control. The latter type requires, for instance, the selection criteria of transactions, the selection criteria of progress attributes, and the actions to be taken to resolve conflicting transaction (e.g. roll-back and abort).

One of the best-known methods drawn from 2PL is the so-called "Dynamic Attributes" (Kumar, 1997). In this approach, when a conflict is detected, the system must recover from the deadlock. The most common solution to such a conflict is to roll back one or more transactions to break the deadlock. In this case, the algorithm yields a list of deadlocked transactions. A decision is needed to be taken on which transaction(s) to roll back in a process known as victim selection. One possible solution is to roll back a transaction that will incur the minimum cost, which is hard to define precisely. This is for minimum cost is determined by a number of factors.

The most important of which is Dynamic Attributes which is based on the creation of Conflict Resolution Set (CRS). This CRS consists of the number of locks acquired so far, number of conflicts occurred so far, number of roll-backs so far, number of blockings occurred so far.

Since Dynamic Attributes acquire their values during execution, they can be used as a measure of the relative progress of a transaction in a concurrent environment, which, in turn, can be used to identify the suitable transaction for resolving conflicts.

Unfortunately, such mechanisms do not provide the optimum conflict resolution. Because Dynamic Attributes approach, resolves conflicts by using pre-defined rigged priorities between each of its dynamic items in a situation full of uncertainty until later stages of the progress of transactions.

We believe that the Dynamic Attributes approach can be enhanced and generalized by taking into consideration the following factors:

A) **Victim selection consideration:** Extending the above CRS with additional factors to enhance victim selection process. This can be achieved by determining:

a. The length of time the transaction has computed, and how much longer the transaction will compute before it computes its designated task.
b. The number of granules the transaction has used.
c. The number of granules the transaction still needs to completes the transaction.
d. and the number of transactions that will be involved in the roll back.

B) **Rollback consideration:** Another important aspect, after choosing victims, is how far this transaction should be rolled back (i.e. either total or partial roll-back to break the deadlock).

C) **Starvation consideration:** In a system where the selection of victims is based primarily on cost factors, it may happen that the transaction itself might be chosen to roll back. Thus, the transaction would be delayed indefinitely yielding a starvation situation. One common solution to avoid the occurrence of starvation is to include the number of times the transaction in question is rolled back in the cost factor.

Our approach avoids favoritism and allows each attribute to participate in the decision of 'how' the conflict should be resolved. Dealing with such a dynamic environment and fuzzy situation makes the utilization of Neuro-Fuzzy approach of prime importance to resolve this problem using fuzzy inference approach. Therefore, a simulation model based on an adaptive Neuro-Fuzzy Conflict

Resolution Inference System (ANF-CRIS) was developed and discussed herein.

Neuro-Fuzzy Expert System Design Goals

A Neuro-Fuzzy system is a fuzzy system that utilizes a learning algorithm derived from or inspired by neural network theory that determines its parameters (fuzzy sets and fuzzy rules) by processing data samples (Jang *et. al*., 1997).

Neuro-fuzzy networks are hybrid systems constructed and integrated using both neural network and fuzzy logic techniques. They possess the advantages of both neural networks (learning abilities, optimization abilities and connectionist structures) and fuzzy systems (human-like and ease of incorporating expert knowledge). Such integration can be categorized into several categories (Cornelius, 1998; Hasegawa *et. al*., 1995; Ishibuchi, 1996; Jang, 1993; Sun, 1994; Jang *et. al*., 1997):

- Fuzzy rule-based systems with learning ability.
- Fuzzy rule-based systems represented by network architectures.
- Neural network for fuzzy reasoning.
- Fuzzified neural networks.
- Adaptive Neuro-Fuzzy systems
- Other approaches.

AN ARCHITECTURAL AND FUNCTIONAL OVERVIEW OF ANF-CRIS

ANF-CRIS resembles a simulation of a fuzzy expert system with a limited explanation/advice facility. It provides a menu interface to its end-users (who are either database or transaction designers). Figure 1 shows an architectural overview of ANF-CRIS, and identifies its major components whose names reflect their functions as follows:

- A rule base containing a number of fuzzy if-then rules.
- A database that defines the membership function of the fuzzy sets used in the fuzzy rules.
- A decision-making unit which performs the inference operation(s) of the active rule.
- A fuzzification interface which transforms the crisp input into degree of match with linguistic values.
- A de-fuzzification interface which transforms the fuzzy results of the interface into crisp output.

The input to the above system are the weights attached to the dynamic attributes for a given transaction, these weights are named after the dynamic attribute holding it. However, if we assume that there are two transactions and three attributes for each transaction, then we can obtain six inputs to the fuzzy inference system. Figure 2 represents a schematic overview of the rule base input and output.

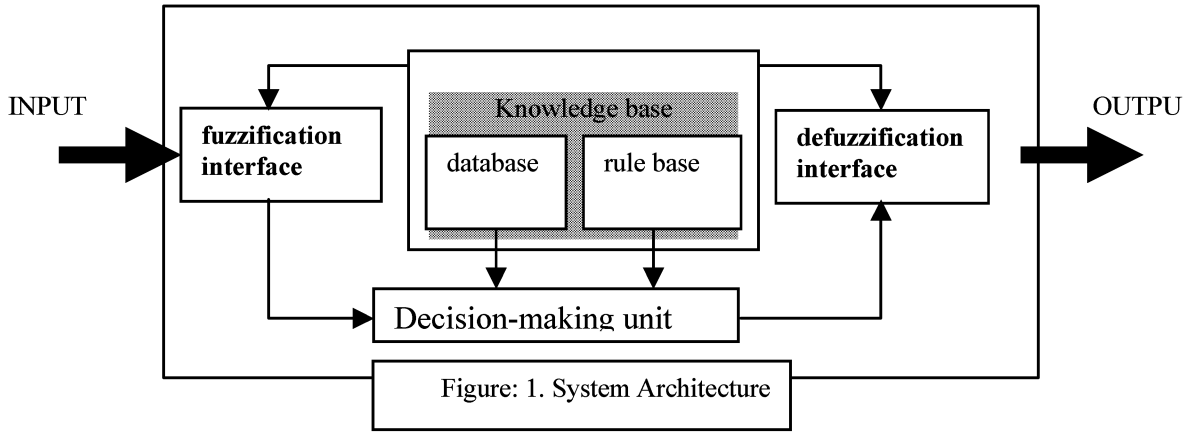


Figure: 1. System Architecture

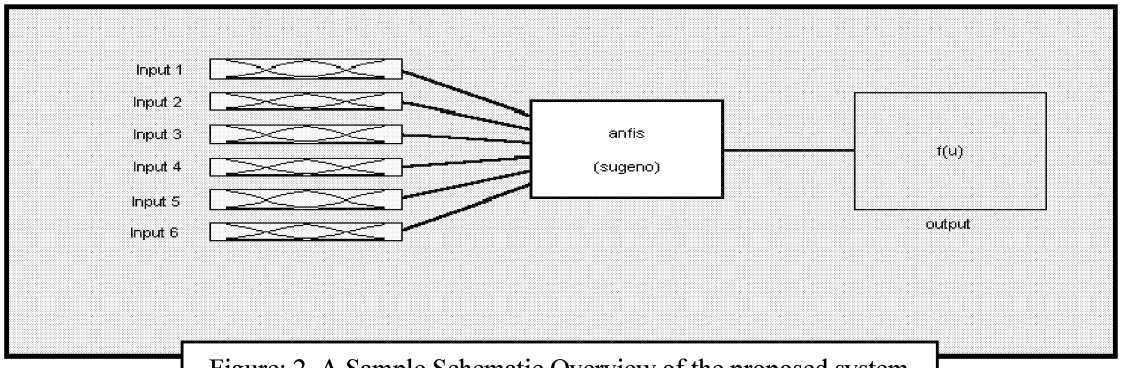


Figure: 2. A Sample Schematic Overview of the proposed system

To illustrate the computation of weights, we choose a simple Conflict Resolution Set (CRS) that consists of only two attributes,
CRS={number of blocks so far, number of conflicts so far}

The output is the priority transaction upon which we decide the elected transactions to be rolled back as shown in Figure 3.

In Dynamic-Attributes weights algorithm, only one attribute is tested at a time, which means that each attribute in the dynamic attribute set has a higher priority than the one that follows. We *generalize* these facts by stating that each attribute has an attached weight, by which we decide the degree of contribution of this attribute when we evaluate the priority of the transaction. We go further to say that priority should also be weighed in order to be able to compare two transactions. Therefore, we re-write the Dynamic Attributes algorithm for the computation of priority in a more coherent way as shown in Figure 4.

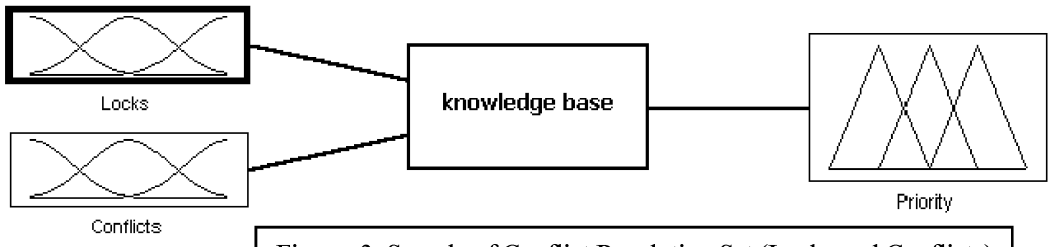


Figure: 3. Sample of Conflict Resolution Set (Locks and Conflicts)

Algorithm Dynamic-Attributes Weights
Let T_i, T_j be two conflicting transactions
Let DAS_i, DAS_j be the dynamic attribute set of T_i and T_j respectively.
$DAS_i=\{a_1,a_2,a_3 \ldots a_n\}$
$DAS_j=\{b_1,b_2,b_3 \ldots b_n\}$
Let P_i, P_k = priorities of T_i, T_j at the time of conflict.
Begin
If $a_1>b_1$ then
Begin
$P_i>P_j$
$b_1=b_1+1$
End
Else
If $a_1=b_1$ then
If $a_2>b_2$ then
Begin
$P_i>P_j$
$b_2=b_2+1$
End
Else
If $a_2=b_2$ then
. . .
If $a_n>b_n$ then
Begin
$P_j>P_i$
$b_n=b_n+1$
End
Else
If $a_n=b_n$ then
$P_i=P_j$
End

Figure: 4. Modified Dynamic Attributes Weight Algorithm

The processing performed by ANF-CRIS when it is applied to a database application consists of two stages, namely the specific knowledge/acquisition stage and the conflict resolution stage. The fuzzy sets are initially defined by the 2PL parameters from the Basic 2PL algorithm on input space. The system is then trained via a special algorithm so that the fuzzy sets are fine-tuned.

Simply, a fuzzy expert system is an expert system that uses a collection of fuzzy membership functions and rules, instead of Boolean logic, to reason about data. The rules in our fuzzy expert system are of a form similar to the following:

Rule 1. If locks is low and conflicts	is low	then priority is low.
Rule 2. If locks is mid and conflicts	is low	then priority is low.
Rule 3. If locks is high and conflicts	is mid	then priority is mid.
...		
Rule n.		

When evaluating priority, the following fuzzy reasoning operations are performed:

- Comparing the input variables with the membership functions, so as to obtain membership values of each rule label (which is usually referred to as fuzzification operation). Each difference is expressed using a membership function. For instance, three fuzzy sets can be used as shown in Figure 5.
- Combining the membership values to get the exact weight for each rule.
- Evaluating the value of the crisp rule; a process that is dependent on the exact weight for each rule.

A *Smaller* membership function means that a given attribute of the first transaction is smaller than the corresponding attribute of the second transaction. Likewise, a larger or equal membership function means that a given attribute of the first transaction is larger than or equal to the corresponding attribute of the second transaction. Since a finite number of attributes are used, three in our case, the third attribute has only two fuzzy sets, smaller and larger. This will ease the decision-making process when the first two attributes are equal.

Also we developed a general model for (ANF-CRIS) input-output mapping as shown in figure 6. If we assume that we have m fuzzy input-output pairs $(\mathcal{X}_p;\mathcal{Y}_p)$, p = 1,2, …, m, where $\mathcal{X}_p = (\mathcal{X}_{p1}, \mathcal{X}_{p2, \ldots,} \mathcal{X}_{pn})$ is an n-dimensional fuzzy input vector and $\mathcal{Y}_p = (\mathcal{Y}_{p1}, \mathcal{Y}_{p2, \ldots,} \mathcal{Y}_{pc})$ is a c-dimensional fuzzy target vector. Therefore, our problem is approximately realize fuzzy mapping from $\mathcal{X}_p$ to $\mathcal{Y}_p$.

The input-output relation of each unit of (ANF-CRIS) with n input units, n_H hidden units, and c output units is written as follows:

Input units: $O_{pi} = \mathcal{X}_{pi}$, $i = 1,2, \ldots, n$,

Hidden units: $Net_{pj} = \sum_{i=1}^{n} W_{ji} \cdot O_{pi} + \Theta_j$,

$j = 1,2, \ldots n_H$,

$O_{pj} = f(Net_{pj})$, $j = 1,2, \ldots, n_H$,

Output units: $Net_{pk} = \sum_{j=1}^{n} W_{kj} \cdot O_{pi} + \Theta_k$,

$k = 1,2, \ldots c$,

$O_{pk} = f(Net_{pk})$, $k = 1,2, \ldots, c$,

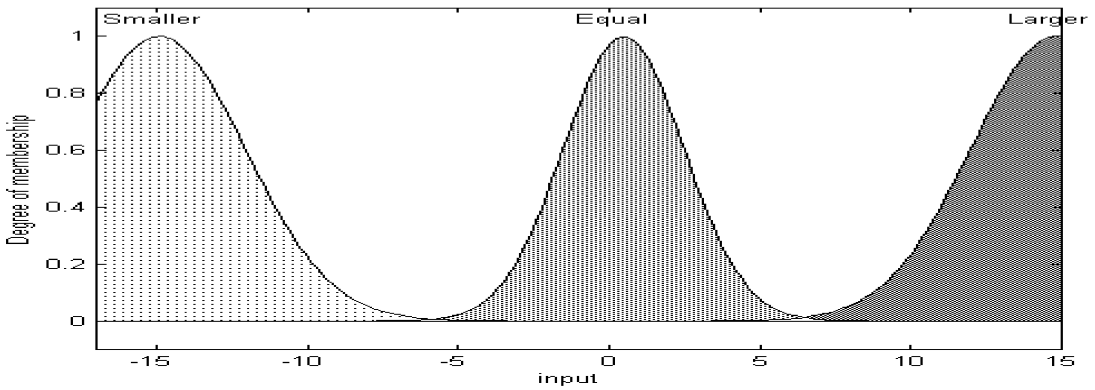


Figure: 5. Membership Function Behavior

In this formulation, the connection weights, W_{ji},W_{kj} and the biases Θ_j, Θ_k are fuzzy numbers.

Let us consider finding a decision consisting of a sequence of hypotheses optimizing some criteria in an environment characterized by available information. Let **D** be a candidate decision consisting of n decision elements d_i , where each decision element di belongs to a finite, descrite set D^i :

D = (*d1, d2, ..., dn), di* D^i.

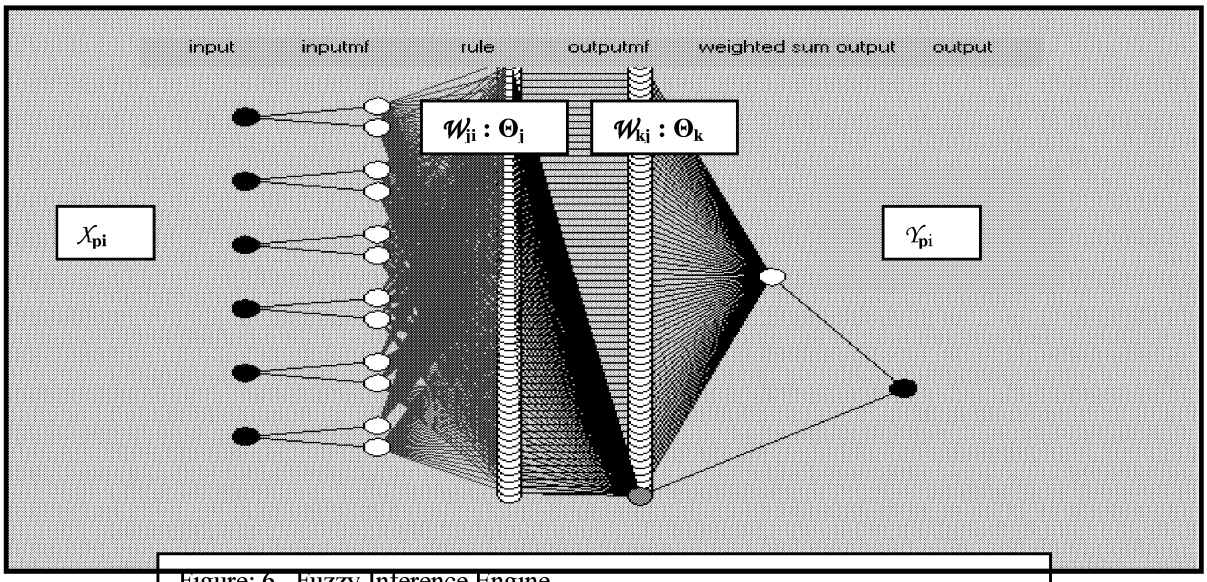


Figure: 6. Fuzzy Inference Engine

As a link between the decision and the available information, a number N of measurements (or observations) are available

$$m_i: D \rightarrow \mathcal{M}: D \rightarrow m_i(D), \; i = 1,\ldots, N,$$

where $\mathcal{M}$ is the measurement space. Heuristic functions are used for rating the different candidate decisions according to these measurements. These ratings describe how well (or how likely) a decision (and its associated measurement) fits in with the environment

$$h_i : \mathcal{M} \rightarrow \mathcal{R} : m_i \rightarrow h_i(m_i) \quad I = 1, \ldots, N,$$

where $\mathcal{R}$ is the space of the possible rating values. Each heuristic can be considered as a piece of knowledge. Heuristics are combined to form a global rating r, which is a measure of the decision

$$r = O[h_1(m_1(\mathbf{D})), h_2(m_2(\mathbf{D})), \ldots, h_N(m_N(\mathbf{D}))]$$

where O is the combination operator across all heuristics. Figure 7 below shows how decision and priority are evaluated for a small sample combination of locks and conflicts.

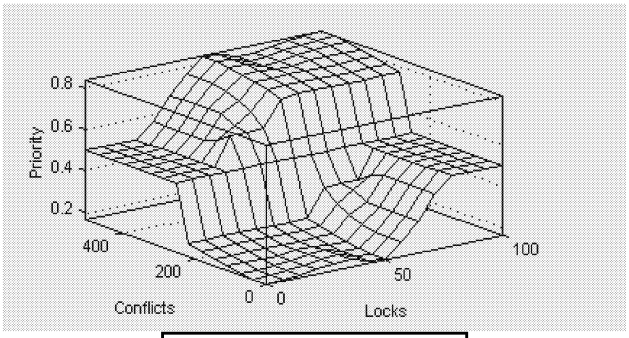


Figure: 7. Priority

SYSTEM OPERATION AND EVALUATION

The basic idea behind system operation is to provide a method for the fuzzy modeling procedure to learn information about a data set. This is essential in order to be able to compute the membership function parameters that best allow the associated fuzzy inference system to track the given input/output data.

The membership function type used in training the system was 'gbellmf' which is shown in Figure 8. The 'gbellmf' is the Generalized Bell Membership Function, GBELLMF(X,PARAMS) returns a matrix which is the generalized bell membership function evaluated at X,PARAMS is a 3-element vector that determines the shape and position of this membership function. Specifically, the formula for this membership function is:

GBELLMF(X, [A,B,C])= 1. /((1+ABS(X-C)/A))^(2*B)).

A training sample of 5000 records was generated using Dynamic-Attributes approach. A random number generator was used in order to generate the attributes of transactions. This was performed to ensure that the system would be trained to a wide spectrum of data.

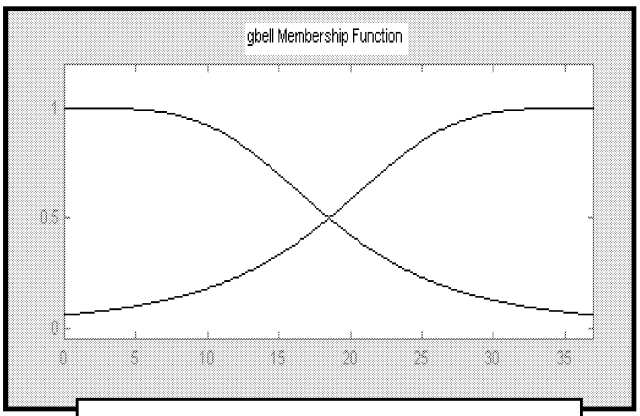
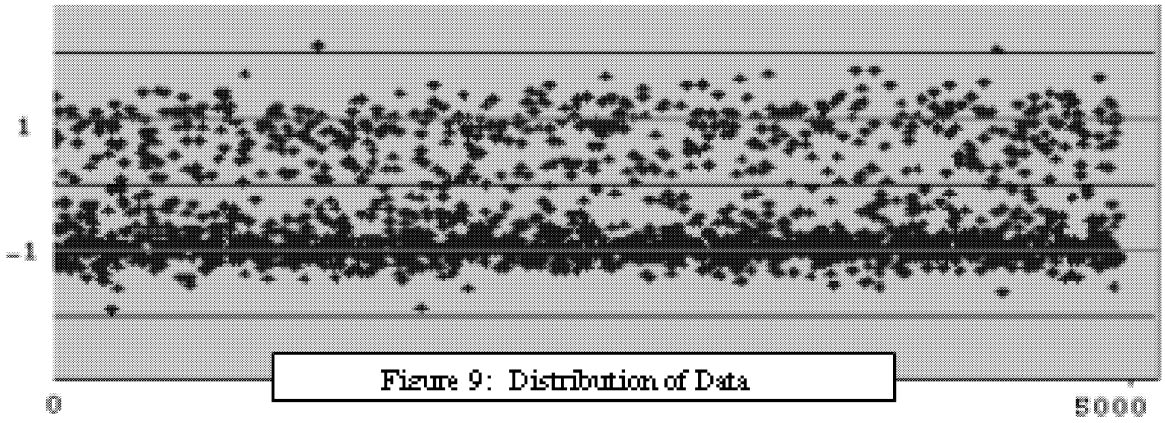


Figure: 8. 'gbell' Membership Function

Each record consisted of six attributes; the first three belonging to the first transaction and the last three belonging to the second transaction. The records were then distributed into two queues representing the first set of transactions (Transaction 1) and the second set of transactions (Transaction 2). These queues were then input to Dynamic-Attributes algorithm.

Dynamic Attributes was modified so that it would select, randomly, a transaction from the first queue and a transaction from the second queue. When the expert system makes its decision of which transaction has higher priority, it make its modifications, and then inserts the modified transactions back into their queues for another cycle, and so on. As a result, during the training of the fuzzy inference expert system, five thousand records were input to the system. Each record consisted of the attributes of transactions selected by Dynamic-Attributes. However, the decision made by the expert system represented by a static value 1 or -1, where 1 indicates that the conflict was resolved in favor of the first transaction, and -1 indicates that the conflict was resolved in favor of the second transaction. Figure 9 shows the distribution of data that our fuzzy inference expert system produced over the values 1 and -1 for one case only.



Fizure 9: Distribution of Data

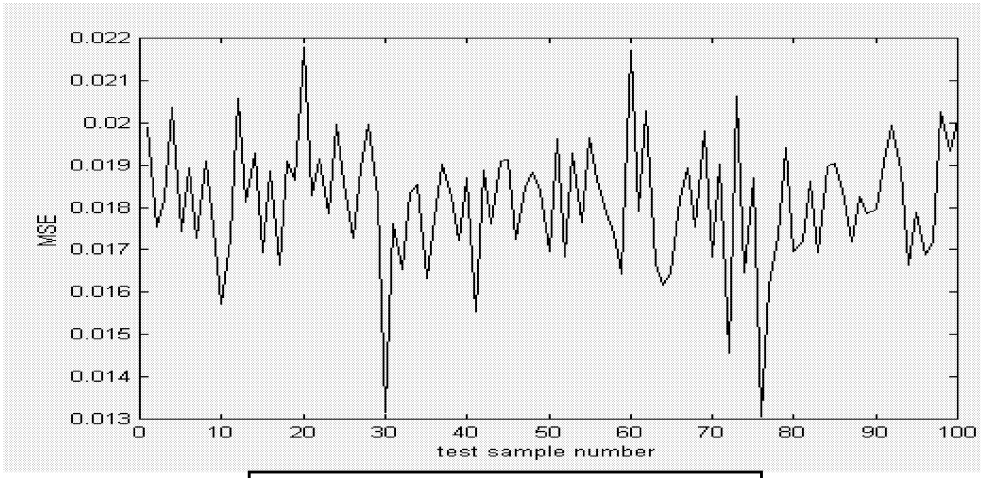


Figure: 10. Summary of MSE for 100 trials

CONCLUSION

We have tested the system for 100 different cases, each of which had 5000 entries as shown in Fig 10 shows MSE for 100 trials.
ANF-CRIS data were compared against the target Dynamic-Attributes data. The overall mean square error (MSE) average was founded to be 0.0181. The error was calculated according to the formula

Error = SUM(ABS(Output of "DA" – Output of FIS))
(Total Number of Input Records)

Where DA is Dynamic-Attributes

The application of an adaptive Neuro-Fuzzy inference approach has been investigated and then developed into the so-called Adaptive Neuro-Fuzzy Conflict Resolution Inference System (ANF-CRIS) for insuring proper transaction conflict resolution. This research demonstrates that Neuro-fuzzy networks, as an adaptive

inference system, is able to resolve transactions conflicts. This is achieved by deciding the degree of contribution of a transaction's attribute in evaluating the priority of that transaction. Our approach avoids favoritism and allows each attribute to participate in the decision of 'how' the conflict should be solved. The fuzzy sets are defined by the parameters from the Basic 2PL algorithm on input space. The processing performed by ANF-CRIS, when it is applied to a database application, consists of two stages, the specific knowledge/acquisition stage and the conflict resolution stage.

The system is applied to functional conflict resolution and compared with its dynamic attributes counterpart. It is superior to the latter in many points including: higher resolution rate, higher prediction ability, insensitivity to overtraining, almost instant training, and its consistency, hence, higher reliability.

REFERENCES

Bernstein, P.; V. Hadzilacos; and N. Goodman. 1987. Concurrency control and recovery in database systems. Addison-Wesley.

Bernstein, P. and E. Newcomer. 1997. Principles of Transaction Processing, Morgan Kaufmann.

Chen, I. 1995. "Stochastic Petri net analysis of deadlock detection algorithms in transaction database systems with dynamic locking", Computer Journal, 38(9), 717-733.

Cornelius, T. L. 1998. Fuzzy logic and expert systems applications, Academic Press.

Elmasri, R. and S. Navathe. 2000. Fundamentals of database systems, Addison-Wesley.

Ermine, J. 1995. Expert systems: theory and practice, Prentice Hall.

Eswaran, K.; J. Gray; R. Lorie; and I. Traiger. 1976. "The notions of consistency and predicate locks in a database system", Communications of the ACM, 19(11), 624-633.

Gray, J. and A. Reuter. 1993. Transaction processing: concepts and techniques, Morgan Kaufmann.

Harder, T. and K. Rothermel. 1993. "Concurrency control issues in nested transactions", VLDB Journal, 2 (1), 39-74.

Hasegawa, T.; S. Horikawa; T. Furuhashi; and Y. Uchikawa. 1995. "On design of adaptive fuzzy controller using fuzzy neural networks and a description of its dynamical behavior", Fuzzy Sets Systems, 71, 3-23.

Haykin, S. 1999. Neural networks: a comprehensive foundation, Prentice Hall.

Ishibuchi, H., 1996. "Development of fuzzy neural networks" In *Pedrycz, Fuzzy Modeling, Paradigms and Practice*, Kluwer Academic, 185-202.

Jang, J., 1993. "ANFIS: adaptive-network-based fuzzy inference system", IEEE Transaction Systems Man Cybernet, 23, 665-685.

Jang, J.; C. Sun; and E. Mizutani. 1997. Neuro-Fuzzy and Soft Computing – A Computational Approach to Learning and Machine Intelligence, Prentice Hall.

Kacprzyk, J. 1997. Multistage Fuzzy Control, John Wiley.

Kumar, V. 1997. "KRISHNA: An efficient concurrency control algorithm based on dynamic attributes of transactions and its performance", Data Knowledge and Engineering, 21(3), 281-296.

Kumar, V. and M. Hsu. 1998. Recovery Mechanisms in Database Systems, Prentice Hall.

Kung, H. and J. Robinson. 1981. "On optimistic methods for concurrency control", ACM Transactions on Database Systems, 6 (2), 213-226.

Kuo, T. and A. Mok. 2000. "Real-time data semantic and similarity-based concurrency control", IEEE Transactions on Computers, 49(11), 1242-1254.

Lee, V. and K. Lam. 2000. "Conflict free transaction scheduling using serialization graph for real-time database", Journal of Systems and Software, 55(1), 57-65.

Lewis, P.; A. Brenstein; and M. Kifer. 2002. Databases and transaction processing – an application oriented approach, Addison-Wesley.

Mak, H. and M. Wong. 2000. "An experimental study of semantics-based concurrency control protocols", Data and Knowledge Engineering, 35(1), 53-81.

Palamidessi, C. (Editor) (2000). *Proceedings of the International Conference on Concurrency Theory*, No. 1877, Springer-Verlag.

Pedrycz, W. 1996. Fuzzy control and fuzzy systems, 2nd Edition, John Wiley.

Reed, D. 1983. "Implementing a tomic actions on decentralized data", Transactions on Computer Systems, 1(1), 3-23.

Russo, M. and L. Jain. 2001. Fuzzy learning and applications, CRC Press.

Silberschatz, A.; H. Korth; and S. Sudarshan. 2002. Database system concepts, McGraw-Hill.

Sun, C. 1994. "Rule-based structure identification in an adaptive-network-based fuzzy inference system", IEEE Transactions Fuzzy Systems, 2, 64-73.

Zadeh, L. 1965. Fuzzy sets. Information and Control, 8, 338-353.

MUNIB QUTAISHAT was born in Swieleh, Jordan and went to Yarmouk University where he studied Computer Science and obtained his B.Sc degree. He got a scholarship from University of Jordan and obtained his M.Sc (1989) and Ph.D. degrees (1993) from U.K. His research interests lie in the area of database systems, software engineering, modeling and simulation, distributed systems and expert systems. Currently he is an associate professor at the Department of Computer Information Systems in the University of Jordan.

MACHINE LEARNING FUZZY METHOD FOR THE MODELLING OF EXPERTS AND SYSTEMS

Peter Otto
Institut für Automatisierungs- und Systemtechnik
Technische Universität Ilmenau
Gustav-Kirchhoff-Straße 1
D-98693 Ilmenau, Germany
Email: peter.otto@tu-ilmenau.de

KEYWORDS
Modelling, machine learning, decision trees, rules, fuzzy system.

ABSTRACT

In this paper a new machine learning method for the production of fuzzy models is presented. The resulting fuzzy models can be used for emulating expert decision behavior or modelling of static as well as dynamic systems. The input – output data collected from the system or expert behavior is converted into linguistic examples using fuzzy sets. The resulting linguistic examples are the basis of the machine learning process for rule production (ID3/C4.5 from Quinlan). The center of gravity method is used for defuzzification and the min-method for the inference machine. The fuzzy sets are optimized in order to minimize the mean square error between the model and the system output. The way this method works is illustrated by a simulation example.

INTRODUCTION

In many sophisticated, hardly comprehensible processes, which for different reasons can not be fully automatized, humans often require very long times, to make competitive decisions.
Frequently such processes are very difficult to model and the determination of data dependencies for diagnosis, monitoring, control/regulation and decision-making is very complicated.
Since the data, which describe the behavior of such processes and the decisions of humans, are often very inaccurate and information can be seized partly from qualitative descriptions, fuzzy systems offer above all a feasible way for information acquisition and processing.
It is possible to represent the relevant dependencies in form of rules, which at the same time contributes the understanding for humans.
Therefore, the task is to optimally reproduce the process dependencies and/or the decisions of humans (fuzzy model).
This task can be solved effectively with methods for automatic knowledge acquisition (machine learning). These include cluster methods (Bezdek 1981), multi-level methods (Vachkov 1994) and the use of methods like ID3 or C4.5 from Quinlan (Srinivasan et al. 1993, Quinlan 1992). They allow the production of rule-sets from large I/O-data sets. As a result, decision trees are produced, which include only significant dependencies as well as minimize the number of rules.

RULE BASED MODELLING

Method and System Description

The decision behavior of an expert and the static and/or dynamic behavior of systems can be described by the following equations:

$$y = f(u_1,u_2,u_3,...,u_n) \quad (1)$$

and

$$y(k) = f(y(k-1),...,y(k-m),u(k-1),...,u(k-n)) \quad (2)$$

where y is the output and $u_1,u_2,u_3,...,u_n$ are the inputs of a static MISO-System,
y(k) is the output and y(k-1),..,y(k-m),u(k-1),..,u(k-n) are the sampled output and input values of a dynamic SISO-System and
f is a non-linear function.

The functional dependencies in equation (1) and equation (2) can be expressed in form of rules, if the input and output values of the system are described by linguistic attributes as a function of the measured values.
In principle, there are two ways to determine the attributes for the inputs.
In the first approach, the attributes are determined by dividing the entire range of the process inputs and outputs into a given number of *n* equal intervals. In the second approach only the output is divided into equal intervals, while a machine learning method optimally specifies the attribute borders for the inputs.
For the fuzzification of the process variables triangular membership functions are assigned to the intervals in such a way that they intersect at the interval borders by $\mu = 0.5$. The edge intervals get one-sid open membership functions (see Figure 1). Thus, a measured value with different membership values belong to two intervals.
The measured values are then transformed according to the

fixed intervals into linguistic attributes, so that a description of the static and/or dynamic expert and/or process behavior in form of linguistic expressions results (Table 1).

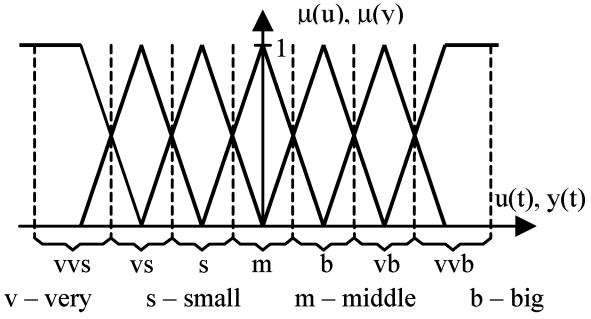


Figure 1: Signal Fuzzification

Table 1: Learning Examples for the ID3-Algorithm

Static	y	u_1	u_2	u_3	u_4
Dynamic	y(k)	y(k-1)	y(k-2)	u(k-1)	u(k-2)
	vvb	vb	b	vb	s
	vvb	vvb	vb	b	vb
	b	vvb	vvb	s	s
	s	b	vvb	vvs	s
	.	.	.	.	.
	.	.	.	.	.
	.	.	.	.	.

The ID3-algorithm (Quinlan 1992) generates an optimal decision tree from the linguistic examples. Production rules of the form

IF (u_1 = vs AND u_3 = b) THEN y = m and
IF (y(k-1) = vb AND u(k-2) = s) THEN y(k) = vvb,

which describe the static or dynamic system/expert behavior, can then be derived from the decision tree.
The decision tree is built in such a way that the information content of the attributes for the decision-making process drops with increasing depth and the irrelevant attributes remain unconsidered.

Generation of the Rule-Set

In analogy to the classical fuzzy design, the data based system design is characterized by an unknown relation between the input and output variables (rule-base). Generation of the rule base is based on the analysis of the existing signals of both input and output variables.
In the automatized design method presented here, the rule base is generated using the well-established ID3 algorithm according to (Quinlan 1992; Otto 1995). In this algorithm, the fuzzy sets characterizing the signal are understood to be signal-to-symbol transformations. Back transformation, i.e. defuzzification, is performed in the opposite direction. The principle of the ID3 algorithm is based on the generation of a decision tree (equations (3), (4)). After starting the procedure with a primary configuration, the generated fuzzy sets are linked in the rules, and the latter are then checked for their "correctness" (mean information content).

$$H(C/a_j) = \sum_{i=1}^{N} P(X_i) H(C/X_i) \Rightarrow \min \qquad (3)$$

This serves to determine those (if ... then...) rules, whose mean information content is maximum.

$$H(C/X_i) = -\sum_{j=1}^{n} P(C_j/X_i) \log_2 P(C_j/X_i) \qquad (4)$$

$H(C/a_j)$ – entropy of occurence of the fuzzy set C_j for the input a_j

$P(C_j/X_i)$ – probability of occurence of the fuzzy set C_j of the output variable, provided that a class X_i of fuzzy sets exists

n – number of fuzzy sets of the output variable

N – number of equivelent classes of examples with the same fuzzy sets being active at the model input

a_j – model input, where uncertainty (entropy) is smallest

In analogy to information processing, the entropy H expresses the expected value, i.e. it may be used as a criterion for evaluating the information content of the information source. After having calculated these minimums, the relations are deleted as are the relations with equivalent classes, where $H(C/X_i) = 0$. This procedure is repeated several times and the relations are combined in the decision tree. The ID3-algorithm infers decision trees by growing them from the root downward, greedily selecting the next best attribute for each new decision branch added to the tree.
ID3 searches a complete hypothesis space (i.e., the space of decision trees can represent any discrete-valued function defined over discrete-valued instances). It thereby avoids the major difficulty associated with approaches that consider only restricted sets of hypothesis: that the target function might not be present in the hypothesis space. The inductive bias implicit in ID3 includes a preference for smaller trees; that is, it searches through the hypothesis space, grows the tree only as large as needed in order to classify the available training examples adequately. Because the training examples are only a sample of all possible instances, it is possible to add

branches to the tree that improve performance on the training examples while decreasing performance on other instances outside this set. Post-pruning permits to avoid overfitting (Mitchell 1997).
Due to the characteristics of this process, not all fuzzy set combinations are considered in the rule base. Consequently, the generated rule base is reduced compared to a complete one, as only those rules are applied, which are really required to simulate the process.
The defuzzification is done using the center of gravity method, whereby the individual terms are determined by the minimum inference method.

Optimization of the Membership Functions

Since, for the production of the rules, homogeneous membership functions were used for all attributes, a further improvement of the fuzzy models by using variable membership functions is expected. In addition, the optimization of the fuzzy sets is done so that the mean square error between model and system output is minimized:

$$Q = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \qquad (5)$$

where: $\hat{y}_i$ – model output y_i – system output
n – number of patterns

For this purpose, the membership functions of the input characteristics are described by four pivot places, so that arbitrary, rectangle, triangle and/or trapezoidal membership functions can be defined.
The other conditions are that maximally 2 attributes are assigned to a measured value and the sum of the membership functions $\mu(u)$ is equal to one. The parameters of these membership functions are therefore given by the start and end points of the intervals of the full membership (roof points $\eta_1, \eta_2 ..., \eta_{2n-2}$). The bottom (foot) points are equal to the points of the roof of the membership functions of the neighboring attributes (see Figure 2).

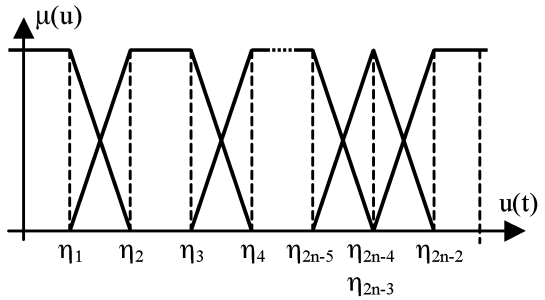


Figure 2: Membership Functions of an Input Variable

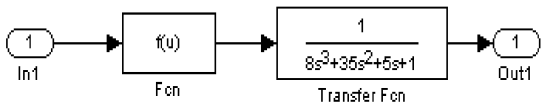
As a result 2(n-1) parameters, $\eta_1, \eta_2 ..., \eta_{2n-2}$ must be optimized for a fuzzy set with n membership functions.

This is a constrained non-linear optimization problem. To solve this problem, the Powell-Method (Powell 1964) as well as the optimization method using evolutionary algorithm proposed by (Schwefel 1977) are applied.
The method, which determines the optimal attribute borders for the inputs, does not change the number of attributes during the optimization process, so that the determined rules are also fully valid after the optimization of the membership functions.
With the method, which divides the inputs into equally large attribute ranges, the allocation of the measured values to the linguistic descriptions is no longer valid after the optimization of the membership functions due to the shift of the originally selected attribute borders, so that the rules determined from it do not have to be optimal likewise any longer. For this reason the entire procedure of the rule determination and fuzzy set optimization must be repeated, until no further improvement in quality can be registered. Hereby, the optimized membership functions of the preceding step are used as initial membership functions for the next learning step. According to experience, 2-4 iterations are sufficient, in order to determine the optimal fuzzy model.

Simulation Results

The above described algorithm for the fuzzy concept was realized in the program system FuzzyOpt (Dung et al. 1997) (equal large attribute ranges for the inputs) and FuzzyMod (optimal attribute borders), which can both be used separately or in combination with the Fuzzy Control Design Toolbox for MATLAB® (Koch et al. 1996). Here the function of the system is going to be demonstrated on an example of a simulated nonlinear dynamic system with Hammerstein-structure:

Simulink®-Model where:
$T_S = 1s$; $-1 \leqslant u \leqslant 1$; ($T_S$ = Sampling time)



non-linearity: $f(u) = 1.36*u-2.06*u^3+1.7*u^5$

Figure 3: Simulated non-linear System

The fuzzy model was determined from a training sequence with 2500 randomly distributed input values with the method, which classifies the model input values optimally and automatically. The number of output classes was preset to 75.
The optimal number of the output classes was determined with an independent test-data set by increasing stepwise the number of classes until the mean square error reached a minimum.
In Figure 4 a comparison of the simulated and the output values estimated by the fuzzy model for an independent test sequence, is presented.

A number of practical applications have shown that the presented identification procedure can be applied to various problems, including modelling the decision behavior of human experts.

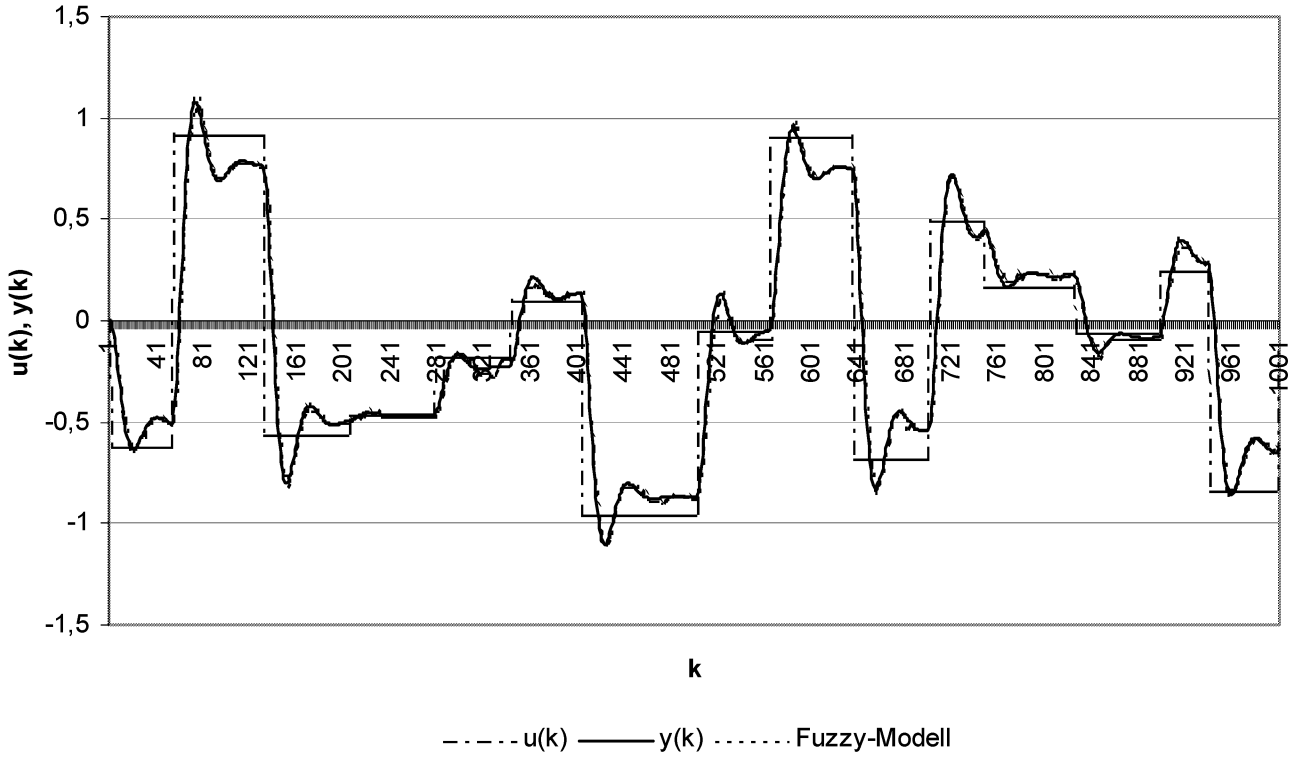


Figure 4: Comparison of the Simulated und Estimated Output Values

A part of the generated rule set is represented in the following:

1. IF (y(k-1) = Term_3_0) THEN (y(k) := Term_0);
2. IF (y(k-1) = Term_3_1) THEN (y(k) := Term_1);
3. IF (y(k-1) = Term_3_2) THEN (y(k) := Term_2);
4. IF (y(k-1) = Term_3_3) THEN (y(k) := Term_0);
5. IF (y(k-1) = Term_3_4) AND (y(k-2) = Term_4_0) THEN (y(k) := Term_3);
6. IF (y(k-1) = Term_3_4) AND (y(k-2) = Term_4_1 OR Term_4_2 OR Term_4_3 OR Term_4_4 OR Term_4_5 OR Term_4_6 OR Term_4_7 OR Term_4_8 OR Term_4_9 OR Term_4_10 OR Term_4_11 OR Term_4_12 OR Term_4_13 OR Term_4_14 OR Term_4_15 OR Term_4_16 OR Term_4_17 OR Term_4_18 OR Term_4_19 OR Term_4_20 OR Term_4_21 OR Term_4_22 OR Term_4_23 OR Term_4_24 OR Term_4_25 OR Term_4_26 OR Term_4_27 OR Term_4_28 OR Term_4_29 OR Term_4_30 OR Term_4_31 OR Term_4_32 OR Term_4_33 OR Term_4_34 OR Term_4_35 OR Term_4_36 OR Term_4_37 OR Term_4_38 OR Term_4_39 OR Term_4_40 OR Term_4_41 OR Term_4_42 OR Term_4_43 OR Term_4_44 OR Term_4_45 OR Term_4_46 OR Term_4_47 OR Term_4_48 OR Term_4_49 OR Term_4_50 OR Term_4_51 OR Term_4_52 OR Term_4_53 OR Term_4_54 OR Term_4_55 OR Term_4_56 OR Term_4_57 OR Term_4_58 OR Term_4_59 OR Term_4_60 OR Term_4_61 OR Term_4_62 OR Term_4_63 OR Term_4_64 OR Term_4_65 OR Term_4_66 OR Term_4_67) THEN (y(k) := Term_1);
:
526. IF (y(k-1) = Term_3_228) THEN (y(k) := Term_74);

Altogether 526 rules were generated. Figure 5 shows a part of the determined optimal membership functions.

CONCLUSION

This paper has shown that fuzzy models can describe nonlinear dynamic systems very well. By using machine learning methods (ID3-algorithm proposed by Quinlan), the declarative knowledge in form of the rule base can be determined problem-free.
A further improvement of the model is possible by an additional optimization of the membership functions. Since, in the method with equally large intervals for the inputs, the arrangement of the measured values to the attributes can change, an iterative approach is necessary in this case. This leads after approx. 2 to 4 steps for the given number of attributes to an "optimal" fuzzy model in the sense of the minimization of the mean square error between model and system output.
An example of the application of the fuzzy modelling method introduced here is given in (Malberg et al. 2002).

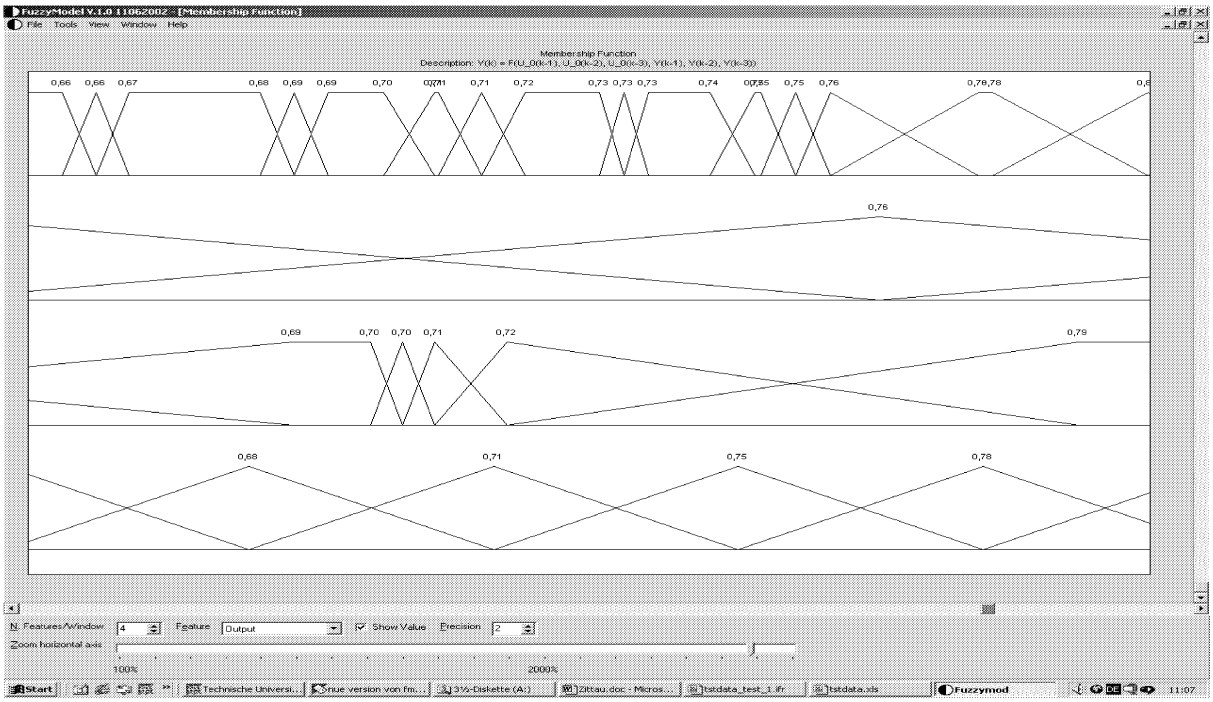


Figure 5: Part of the Determined Membership Functions

REFERENCES

Bezdek, J.C. 1981. *Pattern Recognition with Fuzzy Objective Function Algorithm.* Plenum Press, New York.

Dung, L.T.; M. Koch; and P. Otto. 1997. „FuzzyOpt – ein Werkzeug zum Entwurf optimaler Fuzzy-Systeme." *at – automatisierungstechnik* 45 (1997) 11, 555-556, R. Oldenbourg Verlag.

Koch, M.; Th. Kuhn; and J. Wernstedt. 1996. *Fuzzy Control,* R. Oldenbourg Verlag München Wien.

Malberg, H.; P. Otto; T. Walther; and J. Wernstedt. 2002. „Identifikation der Baroreflex-Dynamik mit Hilfe eines Fuzzy-Modells.", *at – Automatisierungstechnik*, 50 (2002) 05, 212-219, Oldenbourg Verlag München

Mitchell, M.M. 1997. *Machine Learning.* McGraw-Hill Companies, Inc.

Otto, P. 1995. "Fuzzy-Modelling of Nonlinear Dynamic Systems by Inductive Learned Rules". In *Proceedings of the Third European Congress on Intelligent Techniques and Soft Computing*, Vol. 2, 858-864, EUFIT `95, Aachen.

Powell, M.J.D. 1964. "An effecient method for finding variables without calculating derivatives." *Computer Journal* , No. 7, 155-162.

Quinlan, J.R. 1992. *C 4.5. Programs for Machine Learning.* Morgan Kaufmann Publishers, San Mateo, California.

Schwefel, H.P. 1977. *Nichtlineare Optimierung von Computermodellen mittels Evolutionsstrategie.* Birkhäuser Basel und Stuttgart.

Srinivasan, A.; C. Batur; and Ch.Ch.Chan. 1993. "Using Inductive Learning to Determine Fuzzy Rules for Dynamic Systems". *Engng. Applic. Artif. Intell.*, Vol. 6, No. 3, 257-264, Pergamon Press Ltd.

Vachkov, G. 1994. " Mulitlevel Fuzzy Rule Based Modelling". In *Proceedings of the Second European Congress on Intelligent Techniques and Soft Computing*, Vol. 1, 240-245. EUFIT `94, Aachen.

AUTHOR BIOGRAPHY

PETER OTTO was born in Unterpörlitz, Germany. He is an Associate Professor in the Department of System Analysis at the Technical University of Ilmenau, Germany. He received his M.sc. degree in computer science from the Moscow Power Engineering Institut (Technical University), Russia in 1971 and the Ph.D. and the Dr.-Ing. habil. degrees in systems engineering from the Technical University of Ilmenau in 1978 and 1987 respectively. His current research interests are in artificial intelligence techniques applied to system analysis and automatic control problems.

His e-mail address is peter.otto@tu-ilmenau.de.

HIGH PERFORMANCE AND LARGE SCALE COMPUTING

DISTRIBUTED HARDWARE SOFTWARE ENVIRONMENTS

MPI+OPENMP IMPLEMENTATION OF MEMORY-SAVING PARALLEL PIC APPLICATIONS ON HIERARCHICAL DISTRIBUTED-SHARED MEMORY ARCHITECTURES

Sergio Briguglio, Giuliana Fogaccia
and Gregorio Vlad
Associazione EURATOM-ENEA sulla Fusione,
C.R. Frascati,
C.P. 65, 00044, Frascati, Rome, Italy
{briguglio,fogaccia,vlad}@frascati.enea.it

Beniamino Di Martino
Dip. Ingegneria dell'Informazione,
Second University of Naples, Italy
beniamino.dimartino@unina.it
This work has been partly supported by the CNR - Consiglio Nazionale delle Ricerche, Italy (Agenzia 2000 Project *ALCOR* - n. CNRG00A41A).

KEYWORDS

Physics, Multiprocessors, Combined, Hierarchical, Parallel methods.

ABSTRACT

The combination of inter-node and intra-node domain-decomposition strategies for the development of memory-saving parallel Particle-in-cell simulation codes, targeted to hierarchical distributed-shared memory architectures, is discussed in this paper, along with its MPI+OpenMP implementation. Particular emphasis is given to the devised dynamic workload balancing technique.

INTRODUCTION

One of the main issues of thermonuclear fusion research is represented by the investigation of small-scale electromagnetic fluctuations in magnetically confined plasmas and their connection with the high values of the transport coefficients observed in the tokamak experiments (Wesson 1997). The full comprehension of such phenomena requires a self-consistent treatment of the interactions between electromagnetic waves and the plasma, which cannot be satisfactorily described in terms of few moments of the particle distribution function (i.e., by a fluid treatment). A full kinetic approach is instead requested, and significant results cannot be obtained, apart from the case of particular asymptotic regimes, on the basis of analytical studies. Numerical simulation is then needed to get insight in the plasma turbulent behaviour and, in particular, to determine the saturation mechanisms, the level of the saturated fluctuations and their effect on the plasma confinement.

Among the many existing numerical tools, the particle simulation technique (Hockney and Eastwood 1981; Birdsall and Langdon 1985) seems to be the most suited one, as it directly study the time evolution of the particle distribution function and the mutual interaction between such a distribution and the fluctuating electromagnetic fields. Particle simulation indeed consists in performing self-consistent *particle pushing*: the phase-space coordinates of a set of N_{part} particles are evolved in the electromagnetic fields computed, at each time step, in terms of the contribution yielded by the particles themselves (e.g., pressure).

The most important class of particle simulation codes is that of the so-called *particle-in-cell* (PIC) codes, which compute the fluctuating fields and the required particle contribution only at the nodes of a spatial grid, then interpolating the fields at the (continuous) particle position in order to evolve particle coordinates.

Because of the large ratio between the equilibrium scale lengths and the fluctuation ones (typically, several tens or more), high spatial resolution is required in such simulations (up to millions grid cells for three-dimensional PIC codes). Such a requirement, along with the need for ensuring an adequate description of the particle distribution function in the velocity space, makes the usage of large numbers of simulation particles (tens or hundreds millions) necessary and, hence, the full exploitation of parallel computers unavoidable.

Two main workload decomposition strategies have been adopted in parallelizing PIC codes on distributed-memory architectures: the so-called *particle decomposition* (Di Martino et al. 2001b) and *domain decomposition* (Fox et al. 1988; Ferraro et al. 1993). The former strategy statically distributes the particle population among the processes, while assigning the whole domain (and the spatial grid) to each process. The static character of particle assignment yields a perfect load balancing and allows the use of high-level languages like High Performance Fortran (HPF) (High Performance Fortran Forum 1997) in the implementation of this strategy. On the opposite side, an overhead on memory occupancy, given by the replication of data related to the domain, and a computation overhead related to the updating of the fields (each node manages only a partial pressure updating, associated to its portion of particle population) forbid a good scalability of the maximum domain size with the number, n_{node}, of computational nodes. Moreover, though the efficiency of such a technique can be very high, at moderate values of n_{node}, it rapidly degrades when computational loads on each node become dominated by the field-related ones.

Domain decomposition consists instead in assigning different portions of the physical domain and the corresponding portions of the grid to different processes, along with the particles that reside on them. Such a strategy has pros

and cons that are specular to those of particle decomposition. The distribution of all the arrays among the computational nodes gives indeed this method an intrinsic scalability of the maximum domain size (that is, the maximum spatial resolution) that can be simulated with the number of nodes. Particle migration from one portion of the domain to another, however, requires step-by-step particle-to-process reassignment, which can affect the parallelization efficiency, because of the communication and computation overhead. Moreover, it can give rise to serious load unbalances, which require the application of dynamic load-balancing techniques (Cybenko 1989; Ferraro et al. 1993). Both facts make a deep restructuring of the serial code necessary. Moreover, they prevent the use of a high-level data-parallel language and compel one to resort to explicit message-passing libraries, such as MPI.

Such workload decomposition strategies can be extended to the case of shared-memory parallel systems (Di Martino et al. 2001a). In this framework, the use of high-level language, like OpenMP (OpenMP Architecture Review Board 1997), is possible. They can also be combined, when porting a PIC code on a hierarchical distributed-shared memory system (e.g., a cluster of SMPs), in two-level strategies: a distributed-memory level decomposition (among the n_{node} computational nodes), and a shared-memory one (among the n_{proc} processors of each node). In previous papers we have investigated some of these strategies applied to the simulation of thermonuclear plasma confinement. In particular, we have designed and implemented *i*) the particle-decomposition strategy for distributed-memory architectures, with the use of HPF (Di Martino et al. 2001b); *ii*) both particle and domain decomposition strategies for shared-memory architectures, with the use of OpenMP (Di Martino et al. 2001a); *iii*) hierarchically combined particle-particle and particle-domain decomposition strategies for hierarchically structured *distributed-shared memory* architectures, with the integrated use of HPF and OpenMP (Briguglio et al. 2002). The task of optimizing the memory usage requires, however, to enriches this scenario of hierarchically-combined decomposition strategies with the development of the integrated domain-domain combination. Aim of this paper is discussing the MPI+OpenMP implementation of this strategy, with particular emphasis to the dynamic workload balancing technique we have devised.

In the following Sections we describe the inter-node, domain-decomposition strategy, adopted in the distributed-memory context, along with its MPI implementation, and the integration of such inter-node strategy with the intra-node domain-decomposition strategy. Conclusions are drawn in the final Section.

MPI IMPLEMENTATION OF THE INTER-NODE DOMAIN DECOMPOSITION

The typical structure of a PIC code for plasma particle simulation can be represented as follows. At each time step, the code *i*) computes the electromagnetic fields only at the points of a discrete spatial grid (*field solver* phase); *ii*) interpolates the fields at the (continuous) particle positions in order to evolve particle phase-space coordinates (*particle pushing* phase); *iii*) collects particle contribution to the pressure field at the grid points to close the field equations (*pressure computation* phase). We can schematically represent the structure of this time-iteration by the following code excerpt:

```
call field_solver(pressure,field)
call pushing(field,x_part)
call compute_pressure(x_part,pressure)
```

Here, `pressure`, `field` and `x_part` represent pressure, electromagnetic-field and particle-position arrays, respectively. In order to simplify the notation, we will refer, in the pseudo-code excerpts, to a one-dimensional case, while the experimental results reported in the following refer to a three-dimensional (3-D) application.

In implementing a parallel version of the code, according to the distributed-memory domain-decomposition strategy, inter-node communication is required to update the fields at the boundary between two different portions of the domain, as well as to transfer those particles that migrate from one domain portion to another. Such a particle migration possibly determines a severe load unbalancing of the different processes, then requiring a dynamic balancing, at the expenses of further computations and communications.

Three additional procedures then characterize the structure of the parallel code: at each time step, *i*) the number of particles managed by a process has to be checked, in order to avoid excessive load unbalancing among the processes (if such an unbalancing is verified, the load-balancing procedure must be invoked); *ii*) particles that moved from one subdomain to another because of particle pushing must be transferred from the original process to the new one; *iii*) the values of the pressure array at the boundaries between two neighbor subdomains must be corrected, because their local computation takes into account only those particles which belong to the subdomain, neglecting the contribution of neighbor subdomain's particles.

Let us report here the schematic representation of the time iteration performed by each process:

```
 call field_solver(pressure,field)
 call load_checking(i_check,n_part,
&  n_part_left_v,n_part_right_v)
 if(i_check.eq.1)then
  call load_balancing(n_part_left_v,
&   n_part_right_v,n_cell_left,
&   n_cell_right,n_part_left,
&   n_part_right)
  n_cell_new=n_cell+n_cell_left+
```

```
&              n_cell_right
  if(n_cell_new.gt.n_cell)then
   allocate(field_aux(n_cell))
   field_aux=field
   deallocate(field)
   allocate(field(n_cell_new))
   field(1:n_cell)=field_aux(1:n_cell)
   deallocate(field_aux)
  endif
  n_cell=max(n_cell,n_cell_new)
  n_cell_old=n_cell
  call send_receive_cells(field,
&   x_part,n_cell_left,n_cell_right,
&   n_part_left,n_part_right)
  if(n_cell_new.lt.n_cell_old)then
   allocate(field_aux(n_cell_old))
   field_aux=field
   deallocate(field)
   allocate(field(n_cell_new))
   field(1:n_cell_new)=
&       field_aux(1:n_cell_new)
   deallocate(field_aux)
  endif
  n_cell=n_cell_new
  n_part=n_part+n_part_left+n_part_right
 endif
 call pushing(field,x_part)
 call particle_transfer(x_part,n_part)
 allocate(pressure(n_cell))
 call compute_pressure(x_part,pressure)
 call correct_pressure(pressure)
```

In order to avoid continuous reallocation of particle arrays (here represented by `x_part`) because of the particle migration from one subdomain to another, we overdimension (e.g., +20%) such arrays with respect to the initial optimal-balance size, N_{part}/n_{node}. Fluctuations of `n_part` around this optimal size are allowed within a certain oscillation band (e.g., $\pm 10\%$). This band is defined in such a way to avoid a too frequent resort to the time-consuming load-balancing procedure and, at the same time, to prevent an excessive load unbalancing. One of the processes (the MPI rank-0 process) collects, in subroutine `load_checking`, the values related to the occupation level of the other processes and checks whether the band boundaries are exceeded on any process. If this is the case, the "virtual" number of particles (`n_part_left_v`, `n_part_right_v`) each process should send to the neighbor processes to recover the optimal-balance level is calculated (negative values means that the process has to receive particles), and `i_check` is set equal to 1. Then, such informations are scattered to the other processes. These communications are easily performed with MPI by means of the collective communication primitives `MPI_Gather`, `MPI_Scatter` and `MPI_Bcast`. Load balancing is then performed as follows. Particles are labelled (subroutine `load_balancing`) by each process according to their belonging to the units (e.g., the `n_cell` spatial-grid cells) of a finer subdivision of the corresponding subdomain: each particle is then characterized by its unit index and an index ranging from 1 to the number of particles belonging to the same unit. The portion of the subdomain (that is, the number of elementary units) the process has to release, along with the hosted particles, to neighbor subdomains in order to best approximate those virtual numbers (if positive) is then identified. Communication between neighbor processes allows each process to get the information related to the portion of subdomain it has to receive (in case of negative "virtual" numbers). Net transfer information is finally put into the variables `n_cell_left`, `n_cell_right`, `n_part_left`, `n_part_right`. Series of `MPI_Sendrecv` are suited to a deadlock-free implementation of the above described communication pattern.

As each process could be requested, in principle, to host (almost) the whole domain, overdimensioning the grid arrays (`pressure` and `field`) would cause losing of the desired memory scalability (there would be, indeed, no distribution of the memory-storage loads related to such arrays). We then have to resort to dynamical allocation of the grid arrays, possibly using auxiliary back-up arrays (`field_aux`), when their size is modified.

Portions of the array `field` have now to be exchanged between neighbor processes, along with the elements of the array `x_part` related to the particles residing in the corresponding cells. This is done in subroutine `send_receive_cells` by means of `MPI_Send` and `MPI_Recv` calls. The elements of the grid array to be sent are copied in suited buffers, and the remaining elements are shifted, if needed, in order to be able to receive the new elements and to fill possibly occurring holes. After sending and/or receiving the buffers to/from the neighbor processes, the array `field` comes out to be densely filled in the range `1:n_cell_new`. Analogously, the elements of `x_part` corresponding to particles to be transferred are identified on the basis of the labelling procedure performed in subroutine `load_balancing` and copied into auxiliary buffers; the residual array is then compacted in order to avoid the presence of "holes" in the particle-index space. Buffers sent by the neighbor processes can then be stored in the higher-index part of the array `x_part` (remember that such an array is overdimensioned).

After rearranging the subdomain, subroutine `pushing` is executed, producing the new particle coordinates, `x_part`. Particles whose new position falls outside the original subdomain have to be transferred to a different process. This is done by subroutine `particle_transfer`. First, particles to be transferred are identified, and the corresponding elements of `x_part` are copied into an auxiliary buffer, ordered by the destination process; the remaining elements of `x_part` are compacted in order to fill holes. Each process sends to the other processes the corresponding chunks of the auxiliary buffer, and receives the new-particle coordinates in the higher-index portion of the array `x_part`. This is a typical all-to-all communication; the fact that the chunk size is different for each destination process makes the `MPI_Alltoallv` call the tool of choice.

Finally, after reallocating the array `pressure`, subroutine `compute_pressure` is called. Pressure values at the boundary of the subdomain are then corrected by exchanging the locally-computed value with the neighbor process (subroutine `correct_pressure`), by means of `MPI_Send` and `MPI_Recv` calls. The true value is obtained by adding the two partial values. The array `pressure` can now be yielded to the subroutine `field_solver` for the next time iteration.

Note that, for the sake of simplicity, we referred, in the above description, to one-dimensional field arrays. In the real case we have to represent field informations by means of multi-dimensional arrays. This requires us to use MPI derived datatypes as arguments of MPI calls in order to communicate blocks of pages of such arrays.

INTEGRATION OF INTER-NODE AND INTRA-NODE DOMAIN DECOMPOSITION STRATEGIES

The implementation of the domain decomposition strategy for a PIC code at the shared-memory level in a high-level parallel programming environment like OpenMP has been discussed in Refs. (Di Martino et al. 2001a). It consists in further decomposing the node subdomain and assigning a pair of the resulting portions (we will refer to them as to "intervals", looking at the subdivision along one of the dimensions of the subdomain) along with the particles residing therein to each thread. This requires labelling particles according to the interval subdivision. The loop over particles in subroutine `compute_pressure` can be restructured as follows. A pair of parallel loops are executed: one over the odd intervals, the other over the even ones. A loop over the interval particles is nested inside each of the interval loops. Race conditions between threads are then removed from the pressure computation, because particles treated by different threads, will update different elements of `pressure` as they belong to different, not adjacent, intervals. Race conditions can still occur, however, in the labelling phase, in which each particle is assigned, within a parallel loop over particles, to its interval and labelled by the incremented value of a counter: different threads can try to update the counter of a certain interval at the same time. The negative impact of such race conditions on the parallelization efficiency can be contained by avoiding to execute a complete labelling procedure for all the particles at each time step, while updating such indexing "by intervals" only in correspondence to particles that have changed interval in the last time step (Di Martino et al. 2001a).

In integrating the inter-node domain-decomposition strategy with the intra-node one, the most delicate issue is represented by the need of containing the effect of the labelling-procedure race conditions, and, hence, identifying particles whose interval indexing cannot be maintained. Two factors make such a task more complicate in comparison with the simple shared-memory case: the subdomain rearranging (due to load balancing) and the particle migration from one subdomain to another. The former factor may even make the previous-step interval subdivision meaningless; we then choose to reset the interval assignment of particles after each invocation of the load-balancing procedure:

```
...
call send_receive_cells(...)
n_cell=n_cell_new
n_part=n_part+n_part_left+n_part_right
call assign_to_interval(x_part)
...
```

The latter factor enriches the family of particles that change interval: beside those leaving their interval for a different interval of the same subdomain, particles leaving the subdomain or coming from a different subdomain have to be taken into account. In the framework of such domain-domain decomposition strategy, subroutine `particle_transfer` will then include, in addition to the check on inter-subdomain particle migration, the check on inter-interval migration. Particles that left the subdomain will affect the internal ordering of the original interval only; particles who came into the subdomain will be assigned to the proper interval, then affecting only the internal ordering of the new interval; particles that changed interval without leaving the subdomain will continue to affect the ordering of both the original and the new interval.

The analysis aimed to identify, in subroutine `particle_transfer`, inter-subdomain or inter-interval migrating particles can still be performed by a parallel loop over intervals (with a nested loop over interval particles). Race conditions can occur when updating the counters related to particles leaving the subdomain and those related to particles reaching a new interval without changing subdomain. Race conditions can also be presented, of course, when parallelizing the interval assignment of the particles imported from the others subdomains.

Table 1: Elapsed Times (in Seconds) for the Different Procedures of a 3-D Skeleton-Code Implementation of the Decomposition Strategy at Different Pairs n_{node}/n_{proc}, along with the Overall Speed-Up Values

	1/1	*1/2*	*2/1*	*2/2*	*3/1*	*3/2*	*4/1*	*4/2*
Load checking ($\times 10^{-4}$)	0.13	0.13	4.38	4.87	7.20	7.95	11.2	11.0
Pushing	8.52	4.34	4.26	2.14	2.77	1.39	2.10	1.06
Particle transfer	1.27	0.68	0.65	0.36	0.44	0.24	0.33	0.19
Pressure	7.82	3.98	3.85	2.00	2.51	1.34	1.92	1.05
Speed-up	**1.00**	**1.88**	**1.99**	**3.54**	**3.01**	**5.84**	**4.04**	**7.56**

The domain-domain hierarchical decomposition strategy has been implemented for a 3-D skeleton-code. The parallel version of the code has been successfully tested on an IBM SP parallel system, equipped with four 2-processors SMP Power3 nodes, with clock frequency of 200 MHz and 1 GB RAM. Two sources of departures from ideal

speed-up performances can be identified: the computation and communication overheads associated to the sporadic load-balancing procedure (invoked if `i_check.eq.1`) and those related to the every-time-step procedures. Concerning the former source, the execution frequency of that potentially heavy procedure depends positively on the strength of particle dynamics and negatively on the size of the particle-array oscillation band: a larger strength implies a larger migration; a larger size implies a less frequent *load balancing* invocation (but also a larger memory waste). There is apparently a trade-off between parallelization efficiency and memory saving. Note however that the memory waste does not affect the scalability of the problem size with the number of processors, as it is distributed among them. Then we can fix the oscillation-band size in such a way to maintain the load-balancing invocation frequency and its effect on the overall speed-up values at very low levels.

Concerning the latter source of non-ideality, the average elapsed time (in seconds) for the different every-time-step procedures are reported in Table 1 for different pairs n_{node}/n_{proc}, along with the overall speed-up values. Note that the "Pressure" procedure includes both `compute_pressure` and `correct_pressure` subroutines. A case with a spatial grid of $128 \times 32 \times 16$ cells and $N_{part} = 1048576$ particles has been considered.

We note that, for the considered case, the elapsed times decrease with the total number of processors for *pushing* and *pressure* procedures. This is obvious for the former, which is an intrinsically parallel procedure. The latter contains instead the correction phase, which introduces both computation and communication overheads. Such overheads are however overwhelmed, for the considered case, by the standard loop over particles, which, once rearranged in the described nested-loop way, is intrinsically parallel too.

A similar situation occurs for the *particle transfer* procedure, which includes a loop over (all) particles, aimed to identifying migrating particles, and the reassignment phase, reserved to such particles. The former takes advantage from increasing number of nodes and/or processors. The latter becomes relatively more important as these numbers are increased, because of the larger surface-to-volume ratio of each domain portion and the larger corresponding fraction of migrating particles. In the considered case, the benefits are apparently larger than the costs. Such conclusion could be however deeply modified by a stronger particle dynamics and/or for larger number of nodes and processors (i.e., for larger migration rates).

For the *load checking* procedure the elapsed times increase with the number of nodes (the weak dependence on the number of processors has to be considered within the noise level). This is consistent with the n_{node} dependence of both communication and computation loads associated to the corresponding subroutine.

CONCLUSIONS

In this paper we have faced the task of developing a hierarchically-combined domain-domain decomposition strategy for hierarchically structured distributed-shared memory architectures. Such a task is motivated by the need of minimizing the memory requirement, which constitutes one of the major obstacles in performing realistic particle-in-cell plasma simulations. We have discussed an implementation of this strategy based on the combination of MPI, at the inter-node (distributed memory) level, and OpenMP, at the intra-node (shared memory) one. By applying the decomposition strategy to a 3-D skeleton code, we have shown that a good scalability of the maximum simulable domain size with the number of computational nodes can be obtained (as it does not require replicated arrays), along with a satisfactory parallelization efficiency.

A comparison between the four different MPI+OpenMP combined strategies that are obtained by composing particle and domain decompositions at the inter-node and intra-node levels, will be given, for the same skeleton code, in a future paper. The application of these strategies to a real, large scale, 3-D PIC code is in progress.

REFERENCES

Birdsall, C.K. and A.B. Langdon. 1985. *Plasma Physics via Computer Simulation*. McGraw-Hill, New York.

Briguglio, S.; B. Di Martino; and G. Vlad. 2002. "Workload Decomposition Strategies for Hierarchical Distributed-Shared Memory Parallel Systems and their Implementation with Integration of High Level Parallel Languages". *Concurrency and Computation: Practice and Experience*, Wiley, Vol. 14, No.11, 933-956.

Cybenko, G. 1989. "Dynamic Load Balancing for Distributed Memory Multiprocessors". *J. Parallel and Distributed Comput.*, 7, 279-391.

Di Martino, B.; S. Briguglio; G. Vlad; and G. Fogaccia. 2001a. "Workload Decomposition Strategies for Shared Memory Parallel Systems with OpenMP". *Scientific Programming, IOS Press*, 9, No. 2-3, 109-122.

Di Martino, B.; S. Briguglio; G. Vlad; and P. Sguazzero. 2001b. "Parallel PIC Plasma Simulation through Particle Decomposition Techniques". *Parallel Computing*, 27, No. 3, 295-314.

Ferraro, R.D.; P. Liewer; and V.K. Decyk. 1993. "Dynamic Load Balancing for a 2D Concurrent Plasma PIC Code". *J. Comput. Phys.*, 109, 329-341.

Fox, G.C.; M. Johnson; G. Lyzenga; S. Otto; J. Salmon; and D. Walker. 1988. *Solving Problems on Concurrent Processors.* Prentice-Hall, Englewood Cliffs, New Jersey.

High Performance Fortran Forum: High Performance Fortran Language Specification, Version 2.0. 1997. Rice University.

Hockney R.W. and J.W. Eastwood. 1981. *Computer Simulation Using Particles*. McGraw-Hill, New York.

OpenMP Architecture Review Board: OpenMP Fortran Application Program Interface, Version 1.0. October 1997.

Wesson J. 1997. *Tokamaks*. Clarendon Press, Oxford.

Logic Gate Modeling and Simulation using Generalized Discrete Event Specifications : G-DEVS

Aziz Naamane and Norbert Giambiasi ,
LSIS : Laboratoire des Sciences de l'Informatique et des Systèmes, Domaine Universitaire de St Jérôme, Avenue Escadrille, Normandie Niemen, 13397 Marseille Cedex 20, France ;
E-mail: aziz.naamane@lsis.org.

KEYWORDS
GDEVS, Logic gate abstraction ,piecewise linear input output trajectories, discrete event models.

ABSTRACT

Recently, an approach using Generalised Discrete Event Specification paradigm for the simulation of continuous system has been developed. This approach is based on the use of piecewise-linear approximations for the representations of logic gates. The focus of this paper is to show how this technique can be advantageously used for Logic gate modelling and simulation.

INTRODUCTION

In this paper, we propose an original approach to build ***discrete event models*** of logic gates without the classical constraint of piecewise constant input-output trajectories. This allows to build event-driven simulators using piecewise polynomial trajectories as in classical continuous simulators but with a continuous time representation and with an event driven technique in place of a time stepped one. A first work on this kind of approaches was proposed in (Giambiasi 1999;Giambiasi et al. 2000), here, we present a more general approach based on the G-DEVS (Generalized Discrete Event Specification) formalism.
GDEV]S formalism has been introduced recently in the literature to enable the synthesis of accurate discrete event models of highly dynamic continuous processes. GDEVS builds on DEVS (Zeigler 1989--1990;zeigler et al 2000), in that it utilizes arbitrarily higher order polynomial functions for segments instead of the classic piecewise constant segments.
One of the concrete repercussions from the proposed generalization of the classical discrete event abstraction process (to use any kind of trajectories instead of piecewise constant trajectories) is the possibility to simulate continuous or hybrid models with a good approximation by using ***event driven techniques***.
The basic idea for the generalization of the concept of event was introduced for the first time in (Giambiasi et al 1994)] for the purpose of modeling gate delays using fuzzy distributions.
In this paper, we present a method for building discrete event models of logic gate with ***piecewise linear input-output trajectories***.
The remainder of this paper is organized as follows: Section 2 reviews the GDEVS formalism and dynamic system abstraction using the concept of generalized events. Section 3 presents the method for building logic gate models in GDEVS Section 4 describes simulations of an example using the GDEVS simulator. Finally, Section 5 presents some conclusions.

RECALLS

Fundamentally, GDEVS is formalism for the specification of discrete event models of dynamic systems, similar to the formalisms underlying DEVS, Petri nets, automata, logic gate modeling, and hardware description languages such as VHDL and Verilog (Gosh and Giambiasi 1996- -2001;Gosh 1999). The originality of GDEVS stems from the use of polynomials of arbitrary degree, as opposed to constant values, to represent the piecewise input-output trajectories. Thus, in essence, GDEVS constitutes a generalization of the classical discrete event modeling approaches including DEVS, in that a classical model may be viewed as a GDEVS model of order 0. In a GDEVS model ***an event is an instantaneous change in at least one of the values of the coefficients of the polynomial describing the signal***.

GDEVS promises to permit the development of models of greater accuracy while preserving the computational advantages of discrete event simulation. As an example, consider the process underlying a digital system design at different levels of abstraction. At the transistor level, signals may be represented through continuous graphs, as shown in Figure 1a. At the higher, logic gate level, the classical discrete event abstraction employs a

Boolean variable and the signal model utilizes piecewise constant values 0 and 1, as shown in Figure 1b (classical discrete event model). Figure 2 describes the use of piecewise linear approximations of the continuous segments in Figure 1a, which, clearly, represent higher accuracy in modeling the real world behavior. Figure 2 represents a 1-order discrete event abstraction under GDEVS which is computationally faster than the transistor level model in Figure 1a and offers greater accuracy than the classic modeling in Figure1b

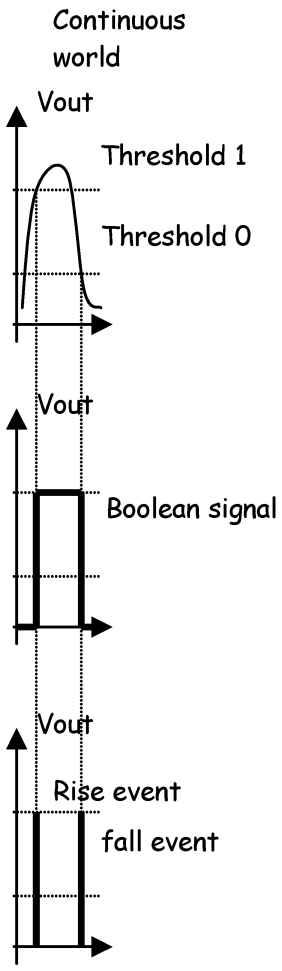


Figure 1a classical Discrete event
Abstraction of a Boolean signal

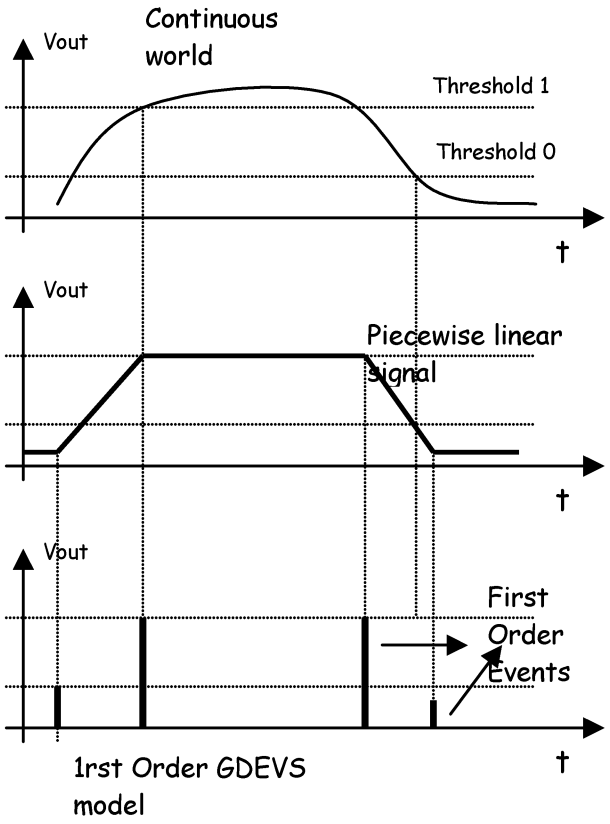


Figure 1b. First order GDEVS abstraction of a signal

LOGIC GATE MODEL : first order GDEVS

To model a logic gate we use three basic components as represented in figure 2: a boolean ideal function, a delay block, and an amplifier block.(Kwok and Preiss 1993).

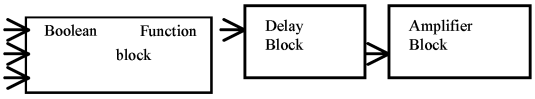


Fig.2 Basic Model Block Diagram

Boolean Function block

This function block is used to model the boolean function of the gate. With a fixed number of inputs, this function computes one output signal. This latter V_{out} is a function of the inputs signals Vi at time t. I.e.:

$$V_{out}(t) = f(V_1(t), V_2(t), \ldots\ldots).$$

For the boolean operators, we have,

And operator : $V_{out}(t) = \min V_i(t)$
Or operator : $V_{out}(t) = \max V_i(t)$
Not operator : $V_{out}(t) = Voffset - V_1(t)$

To build a discrete event abstraction of this continuous function, first order GDEVS is used, so **piecewise linear**

input-output signals are considered, then, an event will be a list of two values (a, b) representing the slope and the gradient of the corresponding linear function. The Boolean operator applies the previous rules to piecewise linear segments defined by the first order events occurring on its inputs. Let us notice that one input event can generate several output events. In general, an input event creates one or several transitory (active) states of the G-DEVS model that implies several output events.

For example , let's consider the following input events (t_i, a_i, b_i) on port 1:
The input events of the first block on port 1 are :
(0,0,0);(3,1,0);(7,0,4);(14,–1,4);(19,1,2);(24,0,7)
(31,–1,7);(34,0,4);(40,0,4)
The input event on port 2 are :
(1,2,0);(4,0,6);(9,0,6);(15,0,0);(20,1,0);(26,0,6);
(35,-1,6);(38,0,3);(40,0,3)

Note that at a conceptual level, the simulation consists in activating the external transition function when an external event occurs, and to activate the output and the internal transition functions when the lifetime duration of the present state is elapsed.
The activation of the internal, external transitions and the output functions is done according to the following pseudo-code :

ATOMIC AND
DECLARATION
INTERFACE
INPUT In[1], INPUT In[2] ;
OUTPUT Out [1]
VARIABLES
Real a_s,b_s , /*output slope and level events */
a_i , b_i , /* input slopes and levels event */
END DECLARATION

EXTERNAL _FUNCTION
$y_1 := a_1*(time-t_1) +b_1$
$y_2 := a_2* (time-t_2) +b_2$

IF $y_1=y_2$ THEN
bs =y_1
IF $a_s< a_1$ and $a_s <a_2$ THEN [
Variable length of lifetime is INFTY
]
ELSE [
IF $a_1<a_2$ THEN [
a_s: = a1
]
ELSE [
$a_s := a_2$

IF$y_1=y_2$ THEN [
bs := y_1
IF $a_s \le a_1$ et $a_s \le a_2$ THEN [
Variable length of lifetime is INFTY
]
ELSE [
IF $a_1 < a_2$ ALORS [
$a_s := a_1$
]
ELSE [
$a_s := a_2$
]
]
]
IF $y_1<y_2$ THEN [
b_s : = y_1
$a_s := a_1$
Variable length of lifetime is INFTY]
]
IF $y_1>y_2$ THEN [
$b_s := y_2$
$a_s := a_2$
[
Variable length of lifetime is INFTY
]
]
IF $a_1 \neq a_2$ THEN [
Crossing event time is calculated t_n :

$$t_n = \frac{a1*t1 - a2*t2 + b2 - b1}{a1 - a2}$$

IF $t_n \le$ time THEN [
t_n := INFTY
]
ELSE[
Crossing level is bn :
$bn = a1*(tn - t1) + b1$
$t_n := t_n$ - time
IF $a_1 < a_2$ THEN [
$a_n := a_1$
]
ELSE [
$a_n := a_2$
]
]
output is sent immediately
]

END EXTERNAL_FUNCTION

INTERNAL_FUNCTION :
/* in case of no external event */
$b_s := b_n$
$a_s := a_n$
output is sent at delta = t_n

END INTERNAL_FUNCTION

OUTPUT FUNCTION :

Pair of coefficients ((a_s, b_s) are sent to output
END_OUTPUT_FUNCTION

END_FUNCTION
END_ATOMIC

The output events at the outport of the first block are :
(0,0,0) ;(3,1,0) ;(7,0,4) ;(11,1,4) ;(15,0,0) ;(20,1,0) ;(26,0,6) ;(32,-1,6) ;(34,0,4) ;(37,-1,4)and (40,0,3)

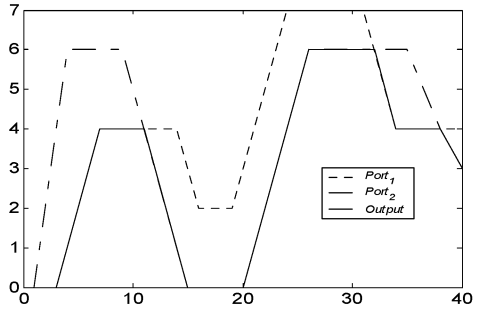


Figure 3 inputs and output of first Block

Delay block

The delay block is used to model the finite, non zero delay of circuit components. In the simplest case, the output of the delay block, V_{delay},is obtained by delaying its input signals, $V_{in}(t)$. In general case, rising signals delay is different from falling signals delay.
Two parameters, characterise the delay block Δr and Δf representing the delay for rising and falling signals (respectively). So, three cases are considered:

- **$\Delta r = \Delta f$** so, **$V_{delay}(t) = Vin(ti-\Delta)$** where $\Delta = \Delta r = \Delta f$
- **$\Delta r < \Delta f$** , so, the delay for rising signals is smaller than the delay for falling signals. Here, **$V_{delay}(t) = \max_{\Delta r < \Delta < \Delta f} V_{in}(t-\Delta)$**
- **$\Delta r > \Delta f$** , in this case, the delay for rising signals is larger than the delay for falling signals. Here, **$V_{delay}(t) = \min_{\Delta f < \Delta < \Delta r} Vin(t-\Delta)$.**

note also that given a piecewise linear continuous input function, the output of the delay block is also piecewise linear continuous function.

Amplifier block

The amplifier block is used to model finite gain, supply rail clipping, and slew rate limiting.
*1) the input signal is multiplied by the gain A, of the amplifier. This is achieved by replacing input events of the form e= (t, (n,b,a)), with events of the form e=(t, (n, Ab, Aa)).
*2) the amplifier block can model power supply rail clipping.in order to respect the minimum output voltage V_{min}, and the maximum output voltage V_{max}.
Given an input signal, $V_{in}(t)$, the clipped voltage is given by

$$V_{clip}(t) = \begin{cases} V_{min} & \text{if } V_{in}(t) < V_{min} \\ V_{in} & \text{if } V_{min} < V_{in}(t) < V\max \\ V_{max} & \text{if } V_{in}(t) > V\max \end{cases}$$

*3) the amplifier block is also used to model limited slew rate (voltage time derivative) of the circuit component. The maximum slew rate of a component is depends on the nominal component's output capacitance, the input capacitance of the devices driven by this component and on the capacitance of the interconnect. Two parameters characterise slew rate limiting : the maximum time derivatives of rising and falling signals δr and δf, (respectively).
Hence, in a slew rate limited event sequence, the a (slope) component of each event satisfies **$\delta f < a < \delta r$.**
The next section explains how a slew limited signal is constructed and shows how to avoid discontinuity like in the case of two consecutive events, e_i (t_i, (n, b_i, a_i,)) and e_{i+1}=(t_{i+1}, (n,b_{i+1}, a_{i+1}))). If **a** is positive and exceeds δr, and if it is simply replaced by δr, then the signal is no longer continuous since $b_i+\delta r(t_{i+1}-t_i) \neq b_{i+1}$.

Slew rate limiting

starting with a sequence of first order events, E(n) ={e_0,e_1,e_2,…} where e_i=(t_i, (n,b_i, a_i)), the aim is to built a new sequence of events, E'(n) = {e'$_0$,e'$_1$,e'$_2$, ….} where e'j=(t_j, (n, w_j,), such that E'(n) is piecewise linear, continuous and slew rate limited,. Computing E'(n) is done according to the method presented in **section 2.1** regarding the specification of Piecewise Linear Trajectories :

$w(t) = a_i.t + b_i \quad \forall t \in <t_i,t_j>$

$w<t_0, t_n> = w<t_0,t_1> * w<t_1,t_2> * . \ldots . w<t_{n-1},t_n>$

thus given e_0=(t_0, (n, b, a)):

At $t=t_0$, $w_0=b_0$. Then for the different piecewise linear segments:

if **$a_i \leq \delta f$ let $a_i=\delta f$** ; if **$ai \geq \delta r$ let $a_i = \delta r$** ; otherwise, **let $a_i = a_i$**
as is noticed slew rate limiting affects only the **a_i** component of signal events.

In our example , the output events of the third block are obtained for a V max=5 V and Vmin =0. The output events of the model are :
(1,0,0);4,1,0);(8,0,4);(12,-1,4);(14,-1,2);(16,0,0);(21,1,0);(27,-1,5);(34,-1,4);(35,0,3); (38,-1,3);(39,0,2)and (41,0,2).

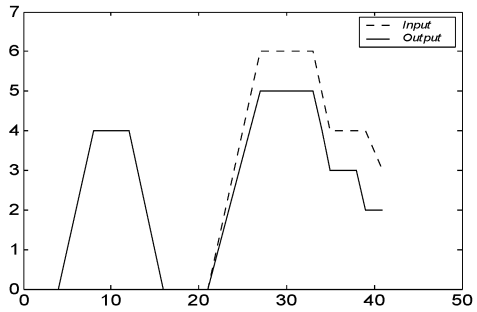


Figure 4 input and output signal of the amplifier block

EXAMPLES:

The proposed approach has been tested by using a software package developed for generalized discrete event simulation, which uses an event oriented simulation kernel and a user oriented simulation language. This declarative language has been developed to enable straightforward descriptions of atomic and coupled GDEVS models.
Several simulations with the three basic gates: and, or and not .were achieved and compared with those obtained in the same conditions by SPICE software (Tuinenga 1998). In this section we give the simulation results for a AND gate
A comparison between simulation results obtained by SPICE and by our approach has been achieved in the same conditions for a NAND gate the following figures give the simulations curves obtained in both cases.
The input event on port1 are :

(0,0,0);(3,1,0);(7,0,4);(14,-1,4);(16,2,0);(19,1,2);(24,7,0),(31,7,-1);(34,0,4);(40,0,4).
The input event on port 2 are :

(1,2,0);(4,0,6);(9,0,6);(15,0,0);(20,1,0);(26,0,6);(35,-1,6);(38,0,3);(40,0,3)

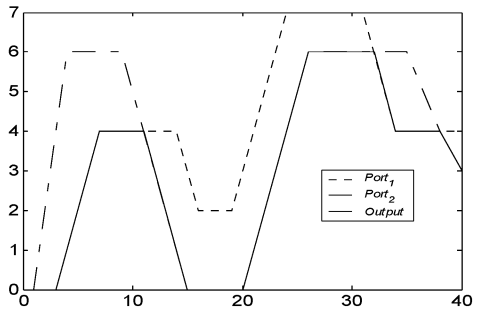


Figure 5 : GDEVS simulation results of AND gate

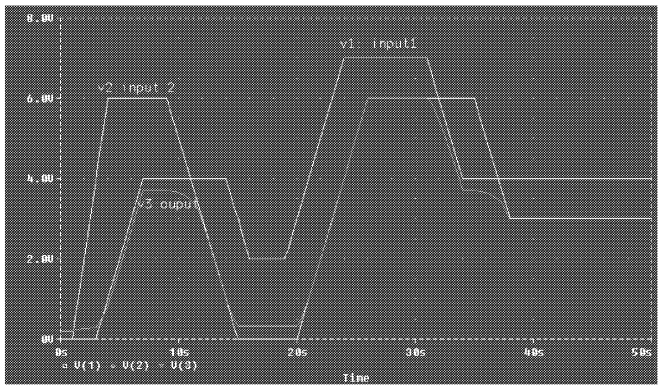


Figure 6 : Spice simulation results of AND gate

The results are very similar except the fact that we notice that in the case of our approach, we don't have the curved lines obtained around 8s and 35s by SPICE software because in our case we use piecewise linear input output segments. The execution time of GDEVS for this example is .8S and for the Spice solution is .11S with a time step 1E-3S, the relative speed advantage of GDEVS is 1.38. It is to be noticed that the SPICE software uses and simulates the gates at the transistor level of abstraction which needs more processing time, which is less faster than in our case. It is to be also emphasized that this comparison is done between a laboratory prototype and an optimized commercial tool.

CONCLUSION

In this paper ,it is presented an approach to gate level timing simulation using Generalised Discrete event Specification , which is based on piecewise linear representations of continuous signals. We are currently developing this approach for others logic components to demonstrate that the potential speed advantages with GDEVS are to be expected. In such cases, as showed by the example discussed above a key advantage of GDEVS is that where higher accuracy is desired, higher order polynomials may be used without any commensurate increase in the number of events. In contrast, in the classical continuous simulation approach, higher accuracies generally require a reduction in the simulation time step which, in turn, implies more execution steps and higher execution time. Thus, the performance advantage of GDEVS over the classical continuous simulation approach is likely to be superior where the use of higher order models is warranted.

REFERENCES

S. Ghosh, N. Giambiasi 1996.
"On the need for consistency between the VHDL language constructs and the underlying hardware design", In Proceedings of 1996 European Simulation Symposium (Genoa, Italy. Oct.16-18), 617-621

S. Ghosh, (1999),
Hardware Description Languages: Concepts and Principles. IEEE Press, , ISBN 0-7803-4744-7.

N. Giambiasi, (1999)
"Abstraction a Evenements Discrets de Systèmes Dynamiques" Communication of , RAIRO APII-Journal Européen des Systèmes Automatisés, N°1(jan.),52-64.

Giambiasi, N., Smaili and Frydman C. (1994). 'Discrete Event Simulation with Fuzzy Delay Models ''. In proceedings of European Simulation Symposium(Istambul.-turkey Sept 14-17) 114-119.

Ghosh, S., Giambiasi.N (2001a)
''Breakthrough in Modeling and Simulation of Mixed-Signal Electronic Designs in n-VHDL", Modeling and Simulation''. In Proceedings of SCSI,(Tuscon-Arizona june 2001) 344-349.

N. Giambiasi, B. Escudé, S.Ghosh (2000a)
"GDEVS: A Generalized Discrete Event Specification for Accurate Modeling of Dynamic Systems".in Proceedings of AIS'2000(Tucson, USA. March 2000) 119-125

B. Zeigler.(1989)
"DEVS representation of dynamical systems", in Proceedings of the IEEE, Vol. 77, 1989, pp. 72-80

B.Zeigler (1990).
"Object-Oriented Simulation with Hierarchical, Modular Models", Academic Press Inc.

B. Zeigler, H. Praehofer and T.G Kim (2000)
" Theory of Modelling and Simulation" - *Academic Press* ISBN 0-12-778455-I.

Tuinenga, P.W (1998)
'' SPICE: A guide to circuit simulation and analysis using PSpice'' Englewood Cliffs,NJ: Prentice-Hall. ISBN 0-13-834607-0

B.T.W.Kwok,B.R Preiss (1993)
'' Simulating continuous systems with piecewise linear signals using time warp''.Research report of University of Waterloo.N2L3GI,waterloo,Ontario,Canada.

DISTRIBUTED COMPUTING FOR ELECTROMAGNETIC OPTIMAL DESIGN

Marco Cioffi, Alessandro Formisano and Raffaele Martone
Dipartimento di Ingegneria dell'Informazione
Seconda Università di Napoli
Via Roma, 29
I-81031, Aversa CE, Italy
marco.cioffi@unina2.it

KEYWORDS
Computational electromagnetics, Optimal design, Genetic algorithms, Cluster computing, Message passing.

ABSTRACT

A survey of the use of parallel computing and other efficient numerical models for computational electromagnetics is presented, showing the particular issues of the implementation of different solution approaches. The adoption of these numerical tools permits to solve many complex and industrial-relevant problems. In particular, the use of parallel Genetic Algorithms for the multi-objective optimal design of electromagnetic devices is illustrated and then applied to the design of Magnetic Resonance Imaging magnets.

INTRODUCTION

Due to their high complexity, the study of actual electromagnetic devices is usually performed by means of suitable numerical tools (Russenschuck 1997). The numerical solution of electromagnetic problems are indeed rather challenging for several reasons (*i.e.* long range interaction, multiple length scale effects). The number of unknown in the adopted mathematical models can be very high, especially when a fine length scale is required. Moreover, to deal with unbounded resolution domains, integral formulations are widely adopted: regrettably, these numerical approaches result in full matrices, often characterised by poor conditioning properties.
Low frequency electromagnetic devices are in addition characterised by complex geometries with a high number of design parameters and by tight constraints. Requirements from consumers and regulatory agencies are looking for more efficient, durable and cheap electrical devices with lower environmental impact. Therefore, to achieve the desired goals, the design parameters should be properly selected on the basis of their impact on the performance figures; in addition. the design of such devices requires the adoption of suitable optimisation schemes, with global search and high efficiency properties.
In the present paper, a general survey of the use of parallel and high performance computing for computational electromagnetics (CEM) is presented. The attention is then focused to the optimal design of superconducting devices, in particular by using a distributed Genetic Algorithm.

EFFICIENT SOLUTION APPROACHES IN CEM

To face with the computational burden arising from the numerical simulation of electromagnetic systems, parallel computing architectures could be strongly recommended together with efficient strategy to distribute matrices among processing nodes and the use of high efficiency numerical libraries for linear algebra, such as BLAS and ScaLAPACK (Rubinstein et al. 2002).
As a matter of fact, the treatment of full matrices, arising from integral formulations, is not efficient on distributed computing architectures due to the heavy inter-processors communication burden. A good approach seems to be the adoption of a hybrid (shared and distributed memory) environment with a cluster of shared memory machines (Ansorge et al. 2000; Manke 2001; Vollaire 1998).
Finite Elements Method (FEM) are widely adopted in CEM: FEM approaches permit today an accurate solution for most engineering and physical problems to be achieved The basic finite element formulation has been developed since the '40 years. The FEM history is outlined in (Zienkiewicz 2000): the increasing problem size and complexity follows the increase in speed and power of computers systems during last years.
Several techniques have been developed to increase the maximum allowable dimension of treatable integral equations problems by adopting different fast methods. These methods allow the computation in $O(N)$ or $O(N \log N)$ operations instead of the $O(N^2)$ operations required by a direct approach. One of the most promising techniques is the Fast Multipole Method (FMM), which allows to reduce the computational cost to $O(N)$ per iteration and to avoid forming dense matrices (Rubinacci et al. 2003). An evolution of this approach is the parallel Multilevel Fast Multipole Method which can be efficiently implemented on cluster machines to solve large scale electromagnetic problems (Velamparambil et al. 1999). By using the parallel Multilevel Fast Multipole Method very large CEM problem has been recently solved up to a dense matrix with 10.2 million unknowns by using a 126 processors cluster machine (Velamparambil et el. 2003).
Also electrical devices optimisation have taken great advantages by using efficient numerical approaches. Stochastic search methods, such as Monte Carlo Analysis (MCA) and Genetic Algorithms (GAs), are widely used for multi-objective optimisation problems for their robustness and straightforward implementation on parallel machines. MCA can be adopted for preliminary studies on the search

space to tune the following optimisation process (Cioffi et al. 2003.

HPC AND PARALLEL COMPUTING FOR ELECTROMAGNETICS

The use of high efficient and powerful parallel and High Performance Computing (HPC) tools is a common practice in many fields of computational physics, such as the computational fluid dynamics (CFD) and other sectors of aerospace industry, and, since '80s, they had begun to spread to the CEM community (Shang 2002), where the experience acquired can give new boost to many research fields (finite elements, coupled problems).
To actually achieve high performances, researchers developing parallel computational methods must consider the connections among the field equations discretisation, the solution algorithm and the parallel computer architecture. For unstructured grids, it is a common practice to adopt domain decomposition to partition the computing tasks among processors. Two of the key issues are to achieve a scalable parallel implementation together with the best single processor performance and load balancing (Manke 2001).
A few years ago most of the supercomputers were massively parallel computers (see for instance the Cray T3E): these machines use the Single Program Multiple Data (SPMD) programming model and they lack any automatic parallelising tool, while the interconnecting network is crucial for parallel performance (Butrylo et al. 2002). In the last years new architectures appeared with hyper nodes composed by a few vector or scalar processors with shared memory and connected by a very high speed network. On these hybrid machines vectorisation and parallelisation tools are available and multiprogramming can be implemented (Multiple Program Multiple Data model, MPMD).
On the other side, the increasing spreading of low-cost and high performances distributed-memory cluster machines is providing to a growing area of researchers the possibility of implement parallel algorithms to computational electromagnetics (Ansorge et al. 2000).
For the different memory architectures specialised strategies have to be adopted to efficiently store assembled matrices arising from the discretisation models (Vollaire et al. 1998).
In CEM a possible approach is to adopt naturally parallel algorithms, such as the domain decomposition, to reduce the size of the problem into different subproblems, together with a multifrontal solver for the linear system of equations (Barton and Rattner 1992).
As previously said, the use of integral Maxwell equations formulations, such as the Boundary Elements Method (BEM) or the integral Finite Elements (FEM) solution, is widely adopted in CEM. Due to the presence of singular integral equation of first kind and long range interactions, the resulting linear systems to solve after the discretisation step are characterised by full and complex coefficients matrices, which are quite challenging to solve for real application 3D problems (Albanese and Rubinacci 1998).
Direct solution methods based on (eventually out-of-core) block LU factorisation are often preferred by industrial users because they are deemed more reliable and robust but they require a price of $O(N^3)$ arithmetic operations. A very used and attractive alternatives to solve these linear systems are the iterative methods, which indeed require always an efficient preconditioning to be effective by changing the spectral properties of the coefficient matrix to increase the convergence rate (Alléon et al. 1997). For non-Hermitian and not positive defined matrices the parallel version of the well known Generalised Minimal Residual Algorithm (GMRES) can be adopted (Velamparambil et el. 2003).
Another possible approach is based on the Element-by-Element scheme together with an iterative solver, such as the Preconditioned Conjugate Gradient (Hsien 2003).

PARALLEL GENETIC ALGORITHMS

The attention will be focused here the optimal design by GAs. The optimal design of a Magnetic Resonance Imaging magnet is discussed in the following and the effectiveness of the multi-populations distributed Genetic Algorithm (Cioffi et al. 2001) is shown.
Genetic Algorithms are members of a class of powerful stochastic, heuristic and adaptive procedures generally called Evolutionary Computation techniques, which are based on the imitation of the genetic processes of biological organisms. GAs may be efficiently used to solve search and multi-objective optimisation problems (Goldberg 1989). These techniques are usually simple and robust and, like other zero-order methods, they do not require computation of the Objective Function (*OF*) derivatives or particular knowledge about *OF* properties. A GA proceeds by creating successive generations of better and better individuals by applying very simple operations. The search is only guided by the fitness value associated to every individual in the population: this value is used to rank individuals depending on their relative suitability for the problem being solved.
The process starts with a randomly generated population of individuals, each one made by strings of the design variables, representing a set of points spanning the search (or design) space. Each individual is suitably coded into a chromosome made by a string of genes: each gene encodes one of the design parameters, by means of a string of bits, a real number or other alphabets. New individuals are then generated by using some genetic operators, the classical ones being the crossover and the mutation. The crossover operator combines two chromosomes to generate an offspring, while the mutation operator randomly changes some of the genes of a chromosome. The probability of survival of the newly generated individuals depend of their fitness: the individuals with higher fitness values are kept in the population for further mating and reproduction and those with lower fitness are discarded.
GAs are generally able to find good solutions in reasonable amounts of time but as they are applied to larger and harder problems, like the electromagnetic optimal design, the time required to find adequate solutions can become very long. Therefore there have been multiple efforts to make GAs

faster and one of the most promising choices is to use parallel implementations (Cantù-Paz and Goldberg 2000). GAs structure is inherently parallel: the simultaneous search for different points in parameters space could be thus naturally distributed among many processes. Different strategies to parallelise GAs have been proposed:

Global Models - These models introduce the parallel computing of new individuals among different processors to speed-up evolution without modifying the main structure and properties of the evolutionary procedure. This method is relatively easy to implement and a significant speed-up can be expected, if the communications cost does not dominate the computation cost. The evaluation of individuals is parallelised by assigning a fraction of the population to each of the processors available. Communication occurs only as each processor receives its subset of individuals to evaluate and when the processors return the fitness values. On a distributed memory computer, the population can be stored in one processor: this "master" processor would be responsible for sending the individuals to the other processors (the "slaves") for evaluation, collecting the results, and applying the genetic operators to produce the next generation. Global models have being used for multi-objective multidisciplinary design optimisation (Mäkinen et al. 1999).

Coarse Grained Models - In these models, also named Island Genetic Algorithms (IGAs), the population of the GA is divided into multiple subpopulations or islands that evolve isolated from each other most of the time, but exchange individuals occasionally. The exchange of individuals is called migration. Coarse grained parallel GAs bring fundamental changes in the operation of the GA and have a different behavior than simple GAs: they introduce new evolution paradigms by using new relations among groups of individuals. Typically, each island explores a different area of the research domain increasing the probability to find a global minimum, but allowing also the possibility of tracking different "quasi-optimal" solutions (local minima): storing multiple quasi-optimal solutions can be useful in optimisation problems, where the possibility of multiple solutions has not to be ruled out, provided that the solutions all satisfy the design objectives. These models are well suited for cluster computing with a little number of nodes connected by a local network.

Fine Grained Models - This approach in parallelising GAs uses fine-grained parallelism. Fine grained parallel GAs partition the population into a large number of very small subpopulations. Indeed, the ideal case is to have just one individual for every processing element available. This model is suited for massively parallel computers.

Hybrid and Multilevel Models - These models concurrently adopt different strategies, for instance with hierarchical multiple levels to perform some sort of meta-optimisation. A possible hybridisation technique can be to combine genetic algorithms with simulated annealing (Ansorge and Shaw 2002): the genetic algorithms to find a selection of attractive local minima, whilst the simulated annealing optimizes the local search.

As already said in IGAs a new migration genetic operator has been included to allow the exchange of individuals among islands. An appropriate migration policy has to be defined among interacting islands: migrations can be 'cooperative', if the islands agree to the migration, or 'dominant', if a more effective island sends its individuals without a previous request. This model is well suited for cluster computing with a limited number of nodes connected by a local network. Note that the evolution of different populations requires the introduction of a sort of "metric" in the chromosomes space, in order to define the "distance" among individuals and populations; in addition global ranking of the various populations must be provided to select the most promising ones.
To increase the efficient use of the computing resources for the resolution of optimisation problems defined by computationally intensive fitness functions, a new population operator has been introduced (Cioffi et al. 2001): the aggression. In the spirit of other GAs concepts, the new operator try to mimic a natural evolution aspect. In natural ecosystems, different species can be viewed as evolutionary islands in competition to gain the limited available resources (*e.g.* food or vital space): the conquest of natural resources by one of the species could be performed for instance by hunting (*the strongest survive*) or resources can be allocated by some kind of "supervising" principle (*the best survive*). In a similar way in the IGA, by adopting the concept of resources allocation, when two islands become too "near", a supervising aggression strategy can decide that the stronger or the more adapt one can conquer the other one by gaining its resources. On the other side, when there is a very weak island and when successive migrations are not able to give new impulse to it, the island can be destroyed and its resources can be given to the strongest island.
The adopted parallel programming paradigm is the message passing, where the parallel running processes communicate, synchronise and exchange data through explicit messages, which have to be set by the programmer. In particular the Single Program, Multiple Data (SPMD) model is used: all processes run the same program, even if the execution of the code may be different among the processes. The message-passing calls follow the Message Passing Interface (MPI) standard.

DESIGN OF MRI DEVICES

As previously recalled, electromagnetic devices are presently called to fulfil stringent and demanding performances: therefore optimal design procedures are often required. Due to the presence of possible quasi-optimal solutions and to the complexity of electromagnetic calculations, the optimal design of electromagnetic devices is a very hard task.
The multi-objective optimal design can be performed by using vector or scalar optimisation techniques. The former are based on the separate evaluation and optimisation of the

different partial objectives, while in the latter the partial objectives are combined in some global cost function (Leyland 2002).
For scalar optimisation the problem is therefore reduced to the minimisation of an Objective Function. The *OF* gives a quantitative value to the satisfaction of the design goals as a function of all design parameters. The quality of each design solution is evaluated with respect to the value of the *OF* in the actual configuration.
Scalar *OFs* are usually assumed as the weighted sum of the different objectives, which have to be properly normalised:

$$OF(\mathbf{x}) = \sum_{i=1}^{k} w_i \, f_i(\mathbf{x}) \qquad (1)$$

where w_i are the weights (with $\sum_i w_i = 1$), k is the number of the partial objectives, f_i is the i-th partial objective normalised in [0,1] and **x** is the design parameters vector. The optimal selection of the weights w_i has to be performed by the designer on the basis of the relative importance of the various objectives and it strongly influences the final results: the weights are an *a-priori* articulation of the designer preferences (Cioffi et al. 2003).
Usually, the scalar *OF* is a non-continuous function and presents multiple local minima, each corresponding to one of the quasi-optimal solutions. When the parameter region of the minimum is unknown, to prevent the local minima attraction the whole parameters space has to be scanned and, therefore, a global search algorithm is recommended (Russenschuck 1997).
To evaluate the practical effectiveness of the parallel GAs for industrial relevant design problems, a case of great applicative interest has been considered, namely the design of a superconducting magnet for Magnetic Resonance Imaging (MRI). The magnets for MRI are designed to provide very strong magnetic fields with quite high levels of homogeneity (Ansorge and Shaw 2002). Suitable optimisation techniques can be effectively used to choose the magnet geometrical parameters (*e.g.* dimensions and position of each coil), while satisfying mechanical and physical constraints (*e.g.* dimensions of the wires and packaging factors or superconductor critical currents) (Cavaliere et al. 2002).
For the design of magnets for MRI, the main performance figure is the homogeneity of the magnetic field inside the spherical Volume of Interest (VOI) of radius R_{VOI}. The lack of homogeneity Unif(**x**) for the magnets configuration **x** is is measured in ppm (parts per million) and is defined as the ratio between the maximum field variation inside the VOI and the central axial field. Due to the analyticity of the field, the points in the VOI where the field get its maximum and minimum values are on the VOI boundary. In the analysis of MRI magnets a spherical harmonics expansion of the axialsymmetric field can be used:

$$B_z = \mu_0 \sum_{n=1}^{\infty} A_n^0 \, R_{VOI}^{n-1} \left[n \cos\vartheta \, P_n^0(\cos\vartheta) + \sin\vartheta \, P_n^1(\cos\vartheta) \right] \qquad (2)$$

where (r, θ) are the cylindrical coordinate of the field point and P^m_n are the Legendre functions of the first kind.

The coefficients A^m_n provide an effective measure of the field homogeneity: for a perfectly homogeneous field all the coefficients are equal to zero except for the first one, which is equal to the field magnitude. Rather effective techniques have been proposed to evaluate A^m_n with good accuracy and limited computation time, allowing to perform analysis with a large number of runs even on a low cost computer (Cavaliere et al. 2002). In addition, the availability of analytical relationships allows to better verify the method convergence, preventing from undesirable effects due to adopting approximated formulae.
The device specifications are usually satisfied by a number of different magnet layouts. If the optimisation procedure is able to provide a number of solution characterised by acceptable "field quality", the designer can have the opportunity to select the final configuration. An important element in the ranking of the solutions could be the robustness against the manufacturing and assembling tolerances, since the geometry of the actual magnet will differ from the nominal one, affecting its performances. Therefore a magnet configuration less sensitive to constructional errors will result more attractive, because it requires lower correction.
The optimal design on MRI devices has been performed by using a global parallel GA model implement on a cluster architecture (Ansorge et al. 2000) and a multi-level hybrid GA (Cioffi et al. 2001).

RESULTS

In this paper the following *OF* has been used for the optimal design of MRI magnets:

$$OF(\mathbf{x}) = w_1 \mathrm{Unif}(\mathbf{x}) + w_2 \mathrm{Vol}(\mathbf{x}) + w_3 \mathrm{Comp}(\mathbf{x}) \qquad (3)$$

where Vol(**x**) is the magnets volume, Comp(**x**) is a measure of the compactness of the device and the coil geometrical parameters are defined in Fig. 1 where a typical magnet poloidal cross section is sketched. It should noticed that the volume of the magnets determines the total amount of used superconductor and therefore it provides an important figure of the device cost. The magnet configuration treated here is composed by 6 superconducting coils. The main goals of the device are a central magnetic field of 3 T with a field homogeneity of 2 ppm over a 10 cm diameter spherical VOI.

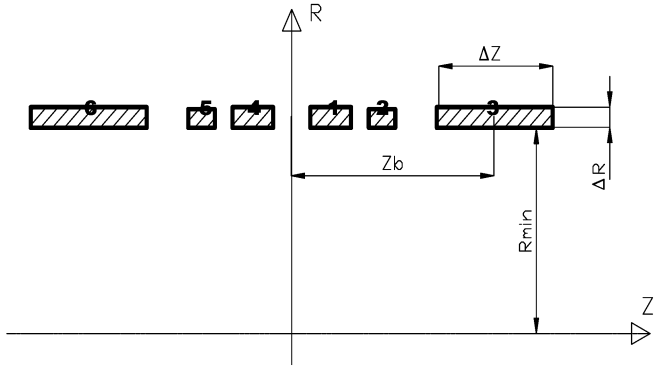


Figure 1: MRI Magnet Poloidal Cross Section

The MRI analysis by IGA has provided a set of different magnets layouts, each one corresponding to a local minimum of the *OF*: the minima parameters are reported in Table 1. The magnet geometry are very similar: one of the difficulties of many engineering optimisation problem is that the local minima can be very close in the search space but often separated by regions of unfeasible solutions.

Table 1: Coils dimensions (in meters) for different optimal magnets

	Magnet A	Magnet B	Magnet C	Magnet D
Rmin	0.2204	0.2155	0.2210	0.2198
Z_b,1	0.0482	0.0582	0.0488	0.0491
ΔZ,1	0.0437	0.0835	0.0446	0.0495
ΔR,1	0.0179	0.0110	0.0190	0.0164
Z_b,2	0.1527	0.1833	0.1536	0.1503
ΔZ,2	0.0551	0.0505	0.0490	0.0352
ΔR,2	0.0159	0.0183	0.0192	0.0251
Z_b,3	0.3019	0.2845	0.3004	0.2932
ΔZ,3	0.0724	0.0723	0.0792	0.0888
ΔR,3	0.0293	0.0209	0.0285	0.0245
OF	**33.71**	**74.95**	**36.32**	**37.00**

CONCLUSIONS

Parallel computing and other HPC techniques are powerful tools to efficiently solve computational electromagnetics problem and their use is rapidly spreading in the scientific and industrial community. The use of the parallel Genetic Algorithms for the optimal design of Magnetic Resonance Imaging devices has been presented: in particular the availability of low-cost cluster machines permits to implement even complex optimisation schemes and to cope with multi-objective complex design tasks.

REFERENCES

Albanese, R. and G. Rubinacci. 1998. "Finite Element Methods for the Solution of 3D Eddy Current Problems". In *Advances in Imaging and Electron Physics*. Academic Press. Vol. 102.

Alléon G.; M. Benzi and L. Giraud. 1997. "Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics". *Numerical Algorithms*. 16(1) (Jan). 1-15.

Ansorge, R.E. and N.R. Shaw. 2002. "Genetic algorithms for MRI magnet design". *IEEE Trans. on Appl. Superconductivity*. 12(1) (March). 733–736.

Ansorge, R.E.; T.A. Carpenter; L.D. Hall; N.R. Shaw and G.B. Williams. 2000. "Use of parallel supercomputing to design magnetic resonance systems". *IEEE Trans. Appl. Superconduct.* Vol. 10. N. 1. 1368–1371.

Barton, M. and J.R. Rattner. 1992. "Parallel computing and its impact on computational electromagnetics". *IEEE Trans. on Magnetics*. Vol. 28. N. 2 (March). 1690 -1695.

Butrylo, B.; F. Musy; L. Nicolas; R. Scorretti and C. Vollaire. 2002. "New trends in parallel electromagnetic fields computation". In *Proceedings of PARELEC '02 International Conference on Parallel Computing in Electrical Engineering* (Warsaw, Poland, Sept. 22-25). 341 -346.

Cantù-Paz, E. and D.E. Goldberg. 2000. "Effcient parallel genetic algorithms: theory and practice". *Computer Methods in Applied Mechanics and Engineering*. 186. 221–238.

Cavaliere, V.; M. Cioffi; A. Formisano and R. Martone. 2002. "Improvement of MRI Magnets Design Through Sensitivity Analysis". *IEEE Trans. Applied Superconductivity*. Vol.. 12. N. 1. 1413-1416.

Cioffi, M.; A. Formisano and R. Martone. 2003. "Stochastic Handling of Tolerances in Robust Magnets Design". In *Proc. Compumag 2003*. (Saratoga Springs, USA, July 13-18).

Cioffi, M.; A. Formisano and R. Martone. 2003. "Uncertainties Treatment in ECT using Parallel Computing Environment". In *Proc. ENDE03* (CEA Saclay, France, May 15-16).

Cioffi, M.; A. Formisano and R. Martone. 2001. "Resources allocation strategy for distributed genetic algorithms: the case of electromagnetic design". In *Proc. SCI 2001 5th World Multiconference on Systemics, Cybernetics and Informatics* (Orlando FL, USA, July 22-25).

Hsieh K. 2003. "Parallelization of EMAP3D based on element-by-element Jacobi preconditioned conjugate gradient method". *IEEE Trans. on Magnetics*. Vol. 39. N. 1 (Jan). 139 -141.

Leyland Geoff. 2002. "Multi-objective optimisation applied to industrial energy problems". *PhD Thesis*. Département de Génie Mécanique. Ecole Polytechnique Fédérale de Lausanne. CH.

Goldberg D.E. 1989. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley. Reading. MA.

Mäkinen, R.A.E.; J. Periaux and J. Toivanen. 1999. "Multidisciplinary shape optimization in aerodynamics and electromagnetics using genetic algorithms". *Intern. J. for Numerical Methods in Fluids*. Vol. 30. N. 2 (May). 149-159.

Manke J.W. 2001. "Parallel computing in aerospace". *Parallel Computing*. Vol. 27. N. 4 (March). 329-336.

Rubinacci, G.; A. Tamburrino; S. Ventre and F. Villone. 2003. "A 3D Fast Multipole Method for Eddy Current Computations". In *Proc. of Compumag Conference* (Saratoga Springs, N.Y. U.S.A., 13-18 July).

Rubinstein, A.; F. Rachidi and M. Rubinstein. 2002. "Development of an optimised parallel numerical electromagnetics code and its implementation on the Swiss T1 and Eridian parallel supercomputers". *18th Annual Review of Progress in Applied Computational Electromagnetics* (Monterey, USA, March).

Russenschuck, S. 1997. "Synthesis, inverse problems and optimisation in computational electromagnetics". *Int. Journal of Numerical Modeling*. Vol. 9. 45-57.

Shang, J.S. 2002. "Shared knowledge in computational fluid dynamics, electromagnetics, and magneto-aerodynamics". *Progress in Aerospace Sciences*. Vol. 38. N. 6-7 (Aug. - Oct.). 449-467.

Shimasaki, M.; T. Iwashita and T. Mifune. 2002. "Fast linear equation solvers in high performance electromagnetic field analysis". *Journal of Computational and Applied Mathematics*. Vol. 149. N. 1 (Dec.). 341-349.

Velamparambil, S.V.; W.C. Chew and J. Song. 2003. "10 Millions Unknowns: Is It That Big?". *IEEE Antennas and Propagation Magazine*. Vol. 45. N. 2 (Apr.). 43-58.

Velamparambil, S.V.; J.E. Schutt-Aine; J.G. Nickel; J.M. Song and W.C. Chew. 1999. "Solving large scale electromagnetic problems using a Linux cluster and parallel MLFMA". *IEEE International Symposium of Antennas and Propagation Society*. Vol. 1. 11-16 (July). 636-639.

Vollaire, C.; L. Nicolas and A. Nicolas. 1998. "Parallel computing for electromagnetic field computation". *IEEE Trans. on Magnetics*. Vol. 34. N. 5 (Sept). 3419 -3422.

Zienkiewicz, O. C. 2000. "Achievements and some unsolved problems of the finite element method". *Int. J. Numer. Meth. Engng*. Vol. 47. 9-28.

DYNAMIC PROBLEM-INDEPENDENT METACOMPUTING CHARACTERIZATION APPLIED TO THE CONDOR SYSTEM

P. Hellinckx, G.Stuer, F. Hancke, D. Dewolfs, F. Arickx, J. Broeckhove and T. Dhaene
Computational Modelling And Programming
Antwerp University
Middelheimlaan, 1 – 2020, Antwerp,
Belgium
E-mail: Pehe@ruca.ua.ac.be

KEYWORDS

Distributed computing, Grid, Condor, Globus, NMI, Benchmarking Classification, Testing.

ABSTRACT

This paper introduces a new approach to the characterisation of Distributed Computational Systems. Basically, the widely used *problem-dependent* benchmarking is replaced by a '*dynamic, problem-independent'* characterisation technique. In this approach a higher level of abstraction is introduced which allows for easy simulation and characterisation of distributed problems. While traditional benchmarking systems start from an existing problem, the proposed technique makes an abstraction of the basic building blocks of these problems, *the tasks.* This is accomplished by replacing them with sleeps of corresponding length. Current research is limited to the simplest possible distribution model in which only CPU-time is taken into account. The current model resembles the characteristics of batch systems, which makes Condor, a well known distributed batch system, a prime candidate for preliminary experiments. These experiments will prove the power and simplicity of the proposed modelling technique.

INTRODUCTION

While in recent times the computational power of modern computing systems has grown spectacularly, the need for increasing computing capacity has grown even faster. A possible solution to this problem is the utilization of the idle CPU-cycles of workstations around the world. Ideally, all unused resources should be made available to whoever needs them. This concept is known as *resource scavenging.* DCS's (Distributed Computational Systems) try to approximate this ideal situation as closely as possible.

Although many DCS's, were (e.g. the Linda Tuple Space implementations, Condor, …) introduced during the last decade(s), there is no actual information on the performance characteristics of those systems. While, all of them are more or less reliable, the key question: *"Which one performs best in this particular situation?"* remains unanswered. Running currently available benchmarks like the KaRMI benchmark suite (Nester et al. 1999) results in static problem-dependent results which do not necessarily correspond to the problem at hand. One only gets exact performance results for the specific problems contained in the benchmark. For all other cases, an error prone interpolation has to be performed.

The goal of this research project is the construction of a classification system which allows one to obtain the performance results closely matching the specific problem at hand without the need for error prone interpolations, and this for any possible DCS. To achieve this, a characterisation has to be made for each DCS. This can be accomplished by a dynamic characterisation mechanism. Such a characterisation is based upon a series of experiments which analyse the influence of different parameters on the performance of the DCS. For now, the scope is limited to the simplest possible distribution model in which only CPU –time is taken into account.

The problem solutions which fall into the scope of this research are characterised by the following parameter tuple: number of tasks, number of workers and the distribution of the task length. In the ideal situation one should be obtain the performance of a given DCS by specifying the parameter tuple without performing any experiment. Because one can simulate each possible problem solution with the proposed technique by supplying its parameter tuple, it is possible to do an empirical detection of the impact of these, mutually dependent, parameters on the overall performance of the DCS. This can be represented as a multi-dimensional parameter space which maps problem solutions to their corresponding performance on a given DCS. This mapping can be performed in two ways, (1) by a brute force approach in which all possible combinations are evaluated, and (2) by a dynamic adaptive statistical approach such as DOE (Montgomery 2001) in which only relevant tuples are selected. As the second method is still ongoing research of a companion project

(Hancke et al. 2003) (De Neve 2003) the experiments described in this paper use the brute force approach.

The following section describes and justifies the proposed testing methodology and shows how the results can be used to reach the characterisation objective. The third section describes the experiments conducted on the Condor batch system and discusses the obtained results obtained.

TESTING METHODOLOGY

Justification

It is easy to justify the introduction of a new testing methodology. One only has to compare the restrictions of the available techniques with the requirements of a characterisation technique.

Restrictions of the current testing techniques:

- **small and limited problem coverage**
  A benchmark is a limited sample of the possible computational problems. One obtains exact performance results but only for the specific problems contained in the benchmark. For all other cases, an error prone interpolation has to be performed.
- **unsatisfactory influence detection**
  Benchmarks usually specialise in detecting the impact of one factor on the performance. Combining the knowledge gained from different benchmarks will lead to a global but imperfect image of the parameters affecting the overhead. The point of failure is that no information is available of the combined effect of the different parameters.
- **system dependent results.**
  Systems are often tuned to reach their optimal performance when running a benchmark. The benchmark results obtained are then no longer representative. As such, they loose their prominent function in DCS characterisation.
- **Strongly depends on the hardware of the workers**
  Our main interest in resource scavenging systems, in which hardware related information like network connection and CPU- power of the available workers is unknown, makes the reflection of hardware dependencies in our test results undesirable noise.

The requirements of a characterisation mechanism:

- **Influence detection**
  Detection of all possible parmeters affecting the DCS's performance.
- **Influence factors dependencies**
  Detection of the dependencies among influencing parameters.
- **Total problem range coverage**
  The ability to simulate every possible computational problem.
- **System independent**
  The ability to test every possible DCS
- **Hardware independent**
  Hardware independent testing is supported to prevent the occurrence of undesirable hardware dependent noise.

Clearly there is a gap between the requirements of a characterisation mechanism and the possibilities of the current testing techniques. Therefore a new testing methodology has to be introduced.

Detailed description of the new approach

The fundamental properties of the proposed testing technique are *total problem range coverage, simulation correctness* and *easy definability*. This means that for every existing distributable problem it should be possible to generate a simulation which is indistinguishable of the original problem and can be completely defined by a couple of parameters. By directly manipulating the fundamental building blocks of distribution problems, the tasks, the properties mentioned above can be obtained. Figure 1 illustrates a typical distribution scheme.

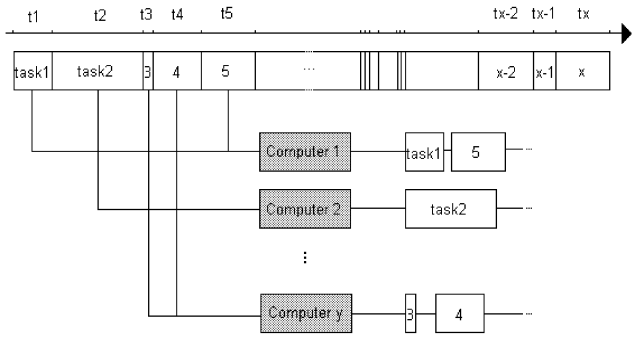


Figure 1: Typical distribution scheme: Each task 'a' takes time 'ta' to compute on one machine. All 'x' tasks are distributed on the 'y' available computers.

Such a scheme consists of 'x' different tasks which can all be computed independently of one another. There is no relation between the tasks and/or the computers that execute them.

Different distributive problems are distinguished by their task lists. Fulfilling the property of having *total problem range coverage* comes down to dealing with this difference in task lists. An easy way to generate every possible task list would solve this problem. This is done by replacing the actual tasks by a time consuming dummy function. See figure. 2.

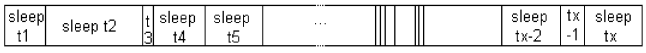


Figure 2: Introducing dummy tasks: each task 'a' is replaced by a sleep of length 'ta'.

This function occupies the processor for a given time. In the current version of our testing technique the tasks are replaced by a sleep function which sleeps the actual running time of the original task (task1 will be replaced by a sleep of time t1). Applying this technique, one can build ery task list one wants by generating the corresponding list of sleeps.
Due to the number of tasks, each with specific properties, in a task list it is impossible to define it manually. To resolve this problem a test-generator is build. Given the appropriate input, an XML-file (McLaughlin 2001) representing the intended task list is constructed. See figure 3.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
<!DOCTYPE tasklist>
    <tasklist>
        <descriptor>
            <nrtasks>10</nrtasks>
            <duration>
                <NormalDistribution std="1.0" avg="10.0" />
            </duration>
        </descriptor>
        <tasks>
            <task id="1">
                <duration>9</duration>
            </task>
            <task id="2">
                <duration>10</duration>
            </task>

                ...

            <task id="10">
                <duration>9</duration>
            </task>
        </tasks>
    </tasklist>
```

Figure 3: Example of an XML file of a taslist with 10 tasks and a normal distribution with mean = 10 and standard deviation = 1.

The use of an XML file has two main advantages:

1. It provides for platform independent storage medium for task lists.
2. Identical experiments can be repeated using the same task list.

Currently the task lists are build using two inputs:

1. The number of tasks.
2. The PDF (probability distribution function) of the task duration.

In this contribution only two PDF's are supported: fixed and Gaussian. In future versions more will be added. Ultimately, a completely user definable PDF will be supported.

Using the test results

Constructing a test is one thing, but drawing definite conclusions is another. The present test design is aimed at investigating the overhead . in the DCS. One has to detect those parameters or combination of parameters that affect this overhead. Possible parameters are:

- The number of tasks.
- The PDF of the task length: The probability function and its descriptors.
  Example: Gaussian distribution, mean, standard deviation.
- The number of workers
- The size in MB of the tasks

In this work we consider the first three parameters only. Given the statistical nature of some of those parameters, enough tests should be run to lead to statistically relevant conclusions.

APPLICATION TO THE CONDOR SYSTEM

What is Condor?

The condor project started in 1988, focusing on customers with large computing needs, and environments with heterogeneous distributed resources. Over the years it evolved towards a distributed batch system where owners have full control over the resources they own. The current Condor system is described best as *"an extensive distributed batch system"*. This definition contains three main concepts, *extensive*, *distributed* and *batch system*, which will be elaborated upon in this section.

The central concept in the definition is "*batch system*". A batch system executes a series of commands, provided by the user in the form of a batch file, without any further interaction of the user. In the Condor case the *distributed* concept has been added.

From the point of view of a user nothing changes, the main principles of a batch system such as the batch file and the absence of user interaction, remain. All changes are internal to the system. Most importantly, the list of tasks will not execute in serial order on the same machine but are distributed to the available workers and executed in parallel. It's obvious that a job implemented in a platform dependent language like C++ or Fortran should be compiled into binaries accepted by the worker's platform, while a job implemented in a portable language like Java only needs the right version of the virtual machine installed.

To take care of the computation, Condor consists of three main components:

1) *The executor :* The machine on which a user requests to run some batch jobs
2) *The workers:* The machines on which the batch jobs are served.

3) *The central manager:* The machine that manages all necessary information of the available workers.

The machines in the pool are a combination of the above components with the constraints that there has to be exactly one central manager and at least one workers and one executor (see Figure 4).

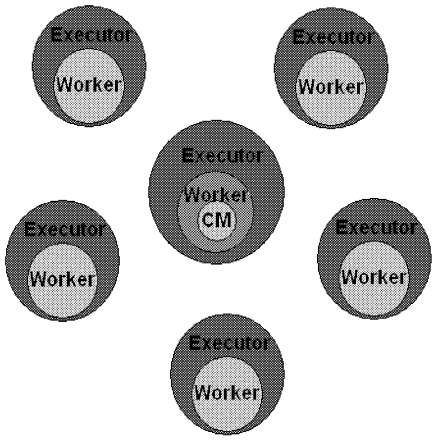


Figure 4: A typical Condor layout: each machine (disks) in the Condor pool can be a combination of a central manager and/or a worker and/or an executor. There has to be exactly one Central manager (CM) and at least one worker and one executor.

The functionality of each component is provided by a number of daemons. The combination of multiple components on one machine is in fact the combination of their daemons.

The Worker component consists of the *master daemon* and the *startd daemon*.

- The *master daemon* is responsible for:
  - The startup of all other Condor daemons.
  - Restarting the other daemons if there is an unnatural daemon exit.
  - Reporting any problem to the administrator by email.
  - Restarting the other Condor daemons if a new binary appears.
- The *startd daemon* is responsible for:
  - Representing a machine to the Condor system.
  - Starting, stopping and suspending jobs according to the owner's policy.

The Executor contains only one daemon: the *schedd daemon*

- The shedd daemon is responsible for:
  - Representing a users to the Condor system.
  - Maintaining the persistent queue of jobs.
  - Contacting the available machines and sending them jobs

The CM (central manager) component consists of the *negotiator* and *collector daemons*.

- The *collector daemon* is responsible for:
  - Collecting information from all other Condor daemons in the pool
  - Each daemon sends a periodic update called a "ClassAd" to the collector
  - Servicing queries for information:
    - Queries from other Condor daemons
    - Queries from users (Condor_status)
- The *negotiator daemon* is responsible for:
  - The matchmaking in Condor
  - Obtaining information from the collector about all available machines and all idle jobs
  - Matching jobs with the machines that will serve them

Finally, the third concept in the Condor definition, *extensive,* will be discussed. Two features will be highlighted here.

The first one is the possibility to communicate with non – Condor grid structures such as Globus which is probably the most important existing middleware toolkit. This offers the possibility of distribution on an enormous scale. It would then theoretically be possible to connect every available computer in the Globus grid to the local condor pool and use the available resources as explained in (Couvares and Todd Tannenbaum 2001).
The second feature is the possibility to set up a reliable and robust resource scavenging system. This is provided by the *master daemon* and *check pointing* . In case of failure, the master daemon can restart any of the other daemons. Check pointing takes a snapshot of the current state of a program in such a way that the program can be restarted from that state at a later time. Due to this, in case of failure, partly executed jobs do not have to be restarted from the beginning, but can be restarted from the last checkpoint.

A good definition of resource scavenging could be *"claiming non-used resources on non-local computers".* In Condor this is accomplished by running the *startd daemon* on the *non-local* machines (workers). This daemon informs the other Condor daemons whether they can use the *non-local* resources or whether the owner of the machine needs the resources himself. Configuring the *startd daemon* is done by writing a number of constraints to a configuration file. The group of possible constraints splits into the job related constraints and the user related constraints. An example of the job related constraints could be that only java jobs are allowed to execute. An example of the job related constraints could be that resources are made available after 10 minutes of mouse inactivity. In summary, the first group specifies *which* jobs can be executed while the second group specifies *when* they can be executed.

Setup of the Condor Tests

Table 1 shows the machines that have been used to perform the tests, as well as their characteristics. Smurf will

distribute the tasks, Drone1of1 till Drone12of1 will serve the tasks and Drone13of1 will function as the Central manager. Every test using x workers will take Drone1of1 till Dronexof1 to work with.

Table 1: Test machines specifications

PC Name	Processor	RAM	OS	Java	Task
Smurf	Intel PIV 1.7 GHz	512 MB	SuSE Linux 8.0	1.4.1	Executor
Drone1of1	Intel PIV 1.7 GHz	256 MB	SuSE Linux 8.0	1.4.1	Worker
Drone2of1	Intel PIV 1.7 GHz	256 MB	SuSE Linux 8.0	1.4.1	Worker
Drone3of1	Intel PIV 1.7 GHz	256 MB	SuSE Linux 8.0	1.4.1	Worker
Drone4of1	Intel PIV 1.7 GHz	256 MB	SuSE Linux 8.0	1.4.1	Worker
Drone5of1	Intel PIV 2.4 GHz	256 MB	SuSE Linux 8.0	1.4.1	Worker
Drone6of1	Intel PIV 2.4 GHz	256 MB	SuSE Linux 8.0	1.4.1	Worker
Drone7of1	Intel PIV 2.4 GHz	256 MB	SuSE Linux 8.0	1.4.1	Worker
Drone8of1	Intel PIV 2.4 GHz	256 MB	SuSE Linux 8.0	1.4.1	Worker
Drone9of1	Intel PIV 2.4 GHz	256 MB	SuSE Linux 8.0	1.4.1	Worker
Drone10of1	Intel PIV 2.4 GHz	256 MB	SuSE Linux 8.0	1.4.1	Worker
Drone11of1	Intel PIV 2.4 GHz	256 MB	SuSE Linux 8.0	1.4.1	Worker
Drone12of1	Intel PIV 2.4 GHz	256 MB	SuSE Linux 8.0	1.4.1	Worker
Drone13of1	Intel PIV 2.4 GHz	256 MB	SuSE Linux 8.0	1.4.1	Central Manager

We also introduce the unambiguous definitions of the quantities that will be considered in comparing the results:

- *Performance P:* The time needed to distribute and execute 'x' tasks of length 'l' on 'y' workers.

- *Overhead OH:* The performance P minus the amount of execution time of each task on the available workers.

$$OH = P - \left\lceil \frac{\#\,\text{tasks}}{\#\,\text{workers}} \right\rceil * tasklength$$

- *Task overhead TOH:* The amount of overhead produced per task.

$$TOH = \frac{OH}{\#\,tasks}$$

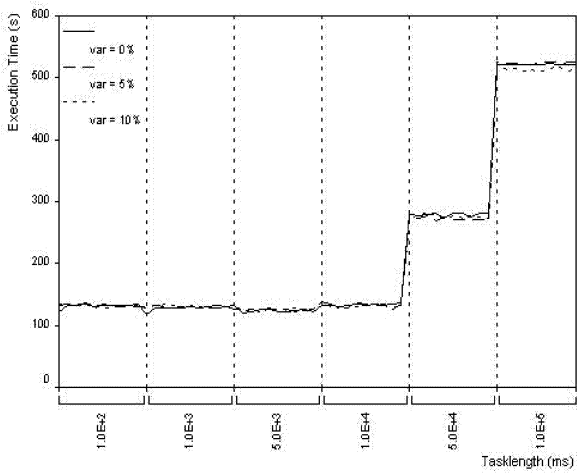
The Condor Tests

The goal of the Condor test is to analyse the impact of the following input factors on the performance:

- Number of workers
- Number of tasks
- The mean length of the tasks
- The standard deviation of the task length

To obtain useful results, an experiment has been conducted for each combination of the following parameters: number of workers 1 to 12; number of tasks 10, 50, 100; task lengths 10, 20, 50, 75, 100, 200, 500, 750, 1000, 2000, 5000, 7500, 10000, 20000, 50000, 75000 and 100000 ms; standard deviation 0, 5, 10 percent. To obtain statistically relevant data, each experiment is executed 10 times.

The initial tests with task lengths of 10, 20 and 50 ms indicated that the standard deviation had no effect on the performance at all. Eliminating this parameter would reduce the number of experiments by a factor of three, so an additional test set was constructed to verify whether this conclusion is valid for all task lengths; it consisted of a combination of the following parameters: number of workers 12; number of tasks 50; task lengths 100, 1000, 5000, 10000, 50000, 100000 ms; standard deviation 0, 5 and 10 percent. For statistical relevance, each test was executed 10 times. Figure 5 indeed shows that the influence of the standard deviation is less than the noise on the resulting data (the small fluctuations in the resulting curves). Because of this irrelevance to the overhead the original test set is reduced tot those experiments with a standard deviation of 0%.



Figures. 5: The total amount of time needed to execute tasklists with given tasklengths and a variance of 0, 5 and 10 % on 12 workers.

The experiments show that the overhead generated by the condor system is strongly dependent upon the task length. The results can be divided in three subsets. The first one, *the overhead subset*, contains all experiments with task lengths less than 1000 ms. The total running time of these experiments is, as his name already suggests, mainly overhead (see Figure 6).

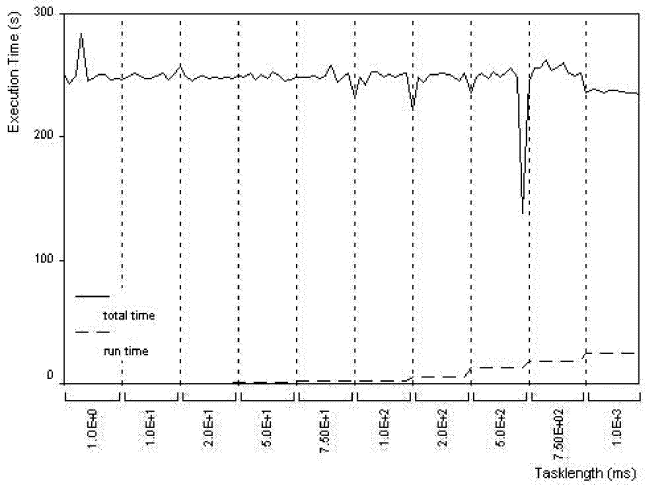


Figure 6: Performance P for the *overhead subset* (task length < 1000ms. The *total time* is the time spent between submitting a task list and retrieving the results. The *run time* is the fraction of the total time that is spent executing the jobs.

The second subset, *the transition subset*, contains experiments with task lengths between 1000 and 50000 ms. It is quite instable and drops from the maximal overhead at task lengths of about 1000 ms to a minimal overhead at task lengths of approximately 50000 ms. Finally the last one, *the performant subset*, contains the remaining experiments with task lengths greater than 50000 ms. As the name suggests this is the most performant subset and the only relevant one. Figure 7 gives a visual impression of the overhead OH in each of the three subsets.

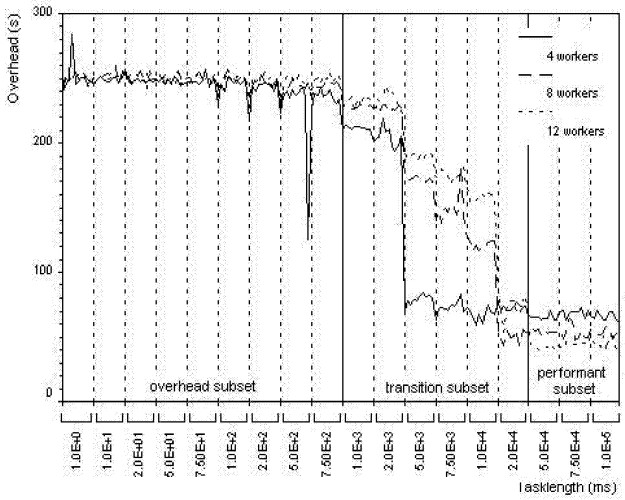


Figure 7. The overhead generated by executing experiments of 100 tasks with a given task length with 4, 8 and 12 workers.

Further analysis of the experiments show, as depicted in figure 8, that distributing a task list is faster than executing the tasks locally when their length is at least 5000 ms. A result of this is that using tasks with lengths in the range of the *overhead subset* is unwise and by consequence irrelevant in the further discussion.

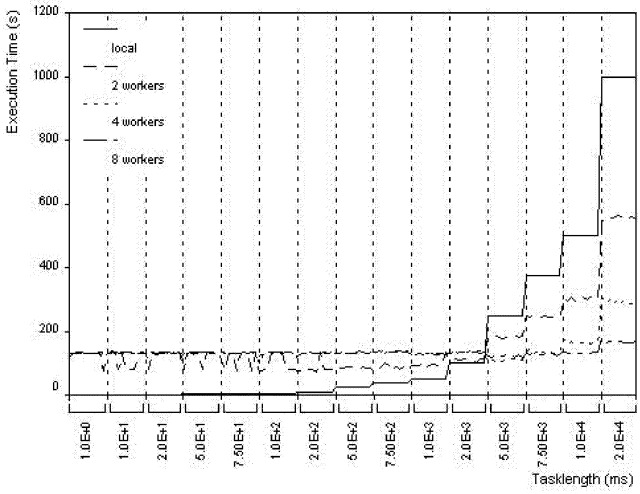


Figure 8: Performance P for up to 8 workers. When the task lengths are larger than 5000 ms, the local execution time, ie. without distribution (full line) is larger than the distributed execution time (dashed lines).

Next the impact of the number of tasks is discussed. Since there is a rather important warm-up cost, the following conclusion are only valid when using a large amount of tasks (>>10).

Analysis shows that the impact of the number of tasks on the task overhead TOH is rather limited. The only noticeable effect is the warm-up cost. Because its contribution to the TOH is inversely proportional to the number of tasks, this factor is only important when the number of tasks is rather small. To prevent the occurrence of multiple warm-ups it is advisable to keep some tasks in the queue at all time. Since there is an overhead associated with each task, it is obvious that OH increases when the amount of tasks increases.

Figure 9 shows that there is an important difference between the tests using one or two workers compared to the tests with more than two workers. First of all the curves for more than two workers are relatively close to each other while the curves for 1 and 2 workers have a completely different shape. More precisely, when using three or more workers, the overhead gradually decreases while for 1 or 2 workers the overhead remains more or less constant. It seems therefore advisable to use one or two workers when the task length is very small and to use more than 2 workers when the task length increases. Because of the irrelevance of small tasks, the use of one or two workers is never really recommendable.

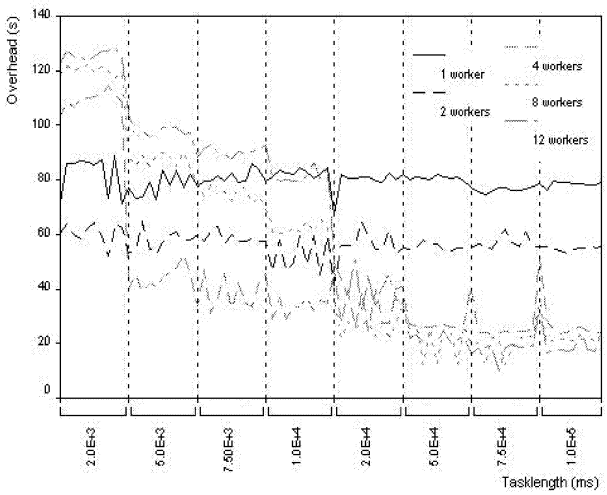


Figure 9: OH for different number of workers. The curves (grey) for more than two workers are relatively close to each other, while the curves (black) for 1 and 2 workers have a completely different shape (rather constant) .

As can be seen from Figure 9, the slope of the curves in the *transition subset* strongly depends on the number of workers. For a small number of workers, the curve shows a discontinuous decrease around a task length of 5000 ms, whereas for a large number of workers a continuous transition from the *overhead* towards the *performant subsets* is obtained. When more than two workers are considered, the TOH reaches its minimum, ± 400 ms with a standard deviation of ± 60 ms, in the *performant subset*. This is caused by the increased distribution of the overhead.

It is important to know whether this trend continues for task lengths of more than 100000 ms. On the other hand, the time needed to execute those tests could take months, so only some key values are tested. We have considered the following additional experiments to test this trend: number of workers: 8; number of tasks: 10, 50, 100; task lengths: 200000, 500000, 750000 and 1000000 ms. As we can see in figure 10, the trend indeed can be extrapolated from the previous conclusions.

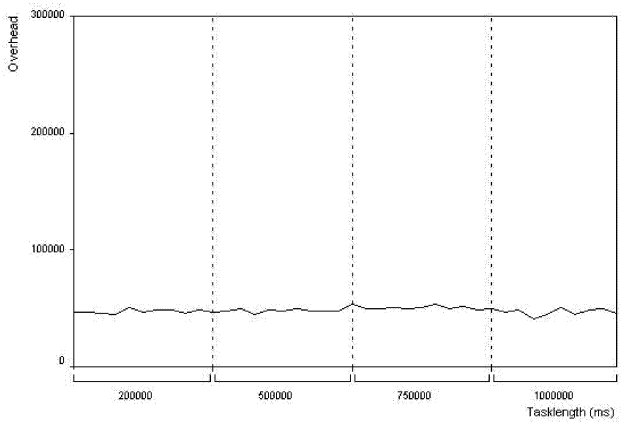


Figure 10. The OH generated by Condor when distributing 100 tasks of 20000, 500000, 750000 and 1000000 ms over 8 workers.

The Condor Conclusions

From the characterisation tests described above, one can conclude that the Condor system is a throughput oriented DCS. These findings correspond with the batch system nature of Condor. Optimal performance is achieved when running many large tasks on a large number of workers. It is advisable to have more than two workers in the Condor pool, to submit tasks which run longer than 50 seconds and to have one or more tasks on the queue at any time. This optimal use will result in a *"Taskoverhead"* of ± 400 ms with a standard deviation of ± 60 ms.
This contribution has shown that introducing a problem independent metacomputing approach to test distributed systems is a valid approach, and leads to predicting conclusions on the performance of these systems. This has been shown using a simple task description model on the specific Condor system.

CONCLUSIONS

In this paper a *problem-independent meta-computing characterization technique* was introduced. It provides a new approach to model a distributed computational systems. The proposed technique enables one to obtain, without performing any experiment, the performance of a problem solution on a given DCS by simply specifying its parameter tuple.
The Condor system was examined using the proposed testing method and was found to be throughput oriented. Condor becomes a viable solution when the lengths of the individual tasks are at least 5 seconds. This is because for smaller task sizes the overhead imposed by the condor system will be larger than the performance gained by the distribution. On the other hand, the Conder system only starts functioning optimal with task lengths of at least 20 seconds. This is because experiments have shown that the overhead drops significantly for tasks larger than 20 seconds.

Future work includes the migration to the dynamic adaptive statistical approach (Hancke et al. 2003) (De Neve 2003) and the refinement of the proposed distribution model by including inter-process communication, processor load, network load, ….

REFERENCES

Chow R. and T. Johnson 1999. "Distributed Operating Systems & Algorithms." 1999. *Addison-Wesley Pub Co* (March 1997)

Couvares, P. and T. Tannenbaum 2001,"Condor Tutorial" *First EuroGlobus Workshop* June 2001

Gropp, W.; L. Ewing and Skjellum A. 1999. "Using MPI". *The MIT Press, second edition*, 1999.

Hancke, F.; G. Stuer; D. Dewolfs; J. Broeckhove; F. Arickx and T. Dhaene 2003. " Modelling Overhead in JavaSpaces". *Proceedings Euromedia* 2003, p77-81

McLaughlin, B..2001. "Java & XML". *O'Reilly & Associates*; 2 edition (September 2001)

Montgomery, D.C. 2001."Design and Analysis of Experiments".the fifth edition. John Wiley and Sons Inc., New York, 2001.

Nester, C.; M.Philippsen and B. Haumacher . "A More Effi-cient RMI for Java". In *Proc. of ACM 1999 Java Grande Conference*, pages 152–157, San Francisco, Calif., June 1999.

Neve, H.D. 2003. "Performantieanalyse van gedistribueerde systemen m.b.v Design Of Experiment"s. 2003. *Thesis UA Academiejaar* 2002 – 2003

Condor: URL: http://www.cs.wisc.edu/condor/

Globus: URL: http://www.globus.org/

NMI: URL: http://www.nsf-middleware.org/

Yale Linda group. URL: http://www.cs.yale.edu/Linda/linda.html

Java Space: URL : http://java.sun.com/products/javaspaces/

Giga Spaces: URL: http://www.j-spaces.com

TSpaces : URL: http://www.almaden.ibm.com/cs/TSpaces/

Xml. URL: http://www.xml.com.

MPICH: URL: http://www-unix.mcs.anl.gov/mpi/mpich

THE HESSE SIMULATION ENVIRONMENT

N. Mazzocca, M.Rak, R. Torella
DII, Seconda Università di Napoli
via Roma 29
81031 Aversa (CE), Italy
+39-081-5010333
n.mazzocca@unina.it
massimiliano.rak@unina2.it ganglio@kaiba.cc

E. Mancini, U. Villano
DI, Università del Sannio
c.so Garibaldi 107
82100 Benevento, Italy
+39-0824-305844
villano@unisannio.it
epmancini@unisannio.it

KEYWORDS

Performance Prediction, Simulation, Heterogeneous Systems, Sweep3d

ABSTRACT

This paper presents a complete component-based simulation environment for Performance Modeking and analysis of Distributed Heterogeneous Computer Systems (DHCS): HeSSE. The HeSSE approach is presented and motivated, and the main components used to model DHCS are described. The use of the simulation environment is shown by means of a case study, the execution of a Sweep 3d Benchmark Code on a SMP cluster. The obtained simulation results are compared to the figures obtained by a simple analytical model and by direct measurement on the target system.

INTRODUCTION

The presence of distributed software systems is pervasive in current computing applications. In commercial and business environments, the majority of time-critical applications has moved from mainframe platforms to distributed systems. In the academic and research field, the advances in high-speed networks and improved microprocessor performance have made clusters or networks of workstations and Computational GRIDS an appealing vehicle for cost-effective parallel computing. More and more complex computer systems implies more and more complex software layers: message passing libraries and environments like PVM and MPI are slowly replaced by complete environment like CORBA, Globus, Web services platforms, which try to make transparent the underlying Operating system and computer architecture to both final users and application developers. We will refer to these environments as Distributed Heterogeneous Systems (DHS).
DHS software design and development has proven to be a challenging matter, due to the inadequacy of the "traditional" approach, commonly followed for sequential or parallel homogeneous systems. Mainframes, supercomputers, workstations or PCs are built on (relatively) stable and well-known hardware architectures and system software, and users can employ customary tools and expertise for software development. The newest DHS systems continuously grow and change, both on hardware and software sides, are subject to subtle effects due to network heterogeneity and sharing.
To face software design complexity in this kind of systems two new branches of software engineering have born: Parallel and Distributed Software Engineering (PDSE), whose landmark was the PDSE Workshop (1996-2000) in the ICSE conference and Software Performance Engineering (WOSP Workshop). PDSE tries to modify Software Engineering techniques and methodologies for applications that require a lot of computation resources, or that are naturally distributed. Software Performance Engineering (SPE) focuses on the methodologies and techniques for designing application to gather better performance.
PDSE projects and studies pointed out some of the main issues in software development for Parallel and Distributed Systems. They can be summarized as follows:

- Definition of clear system life cycle.
- Methodologies for system design.
- Languages for system behavior description.
- Independency of system design from implementation details (as programming paradigms or system architecture)
- Tools for retrieving qualitative and quantitative information about system behavior.

SPE asserts two main principles:

- Performance analysis process should take place at early development stages.
- Analysis should be quantitative and not only qualitative.

Adopting quantitative analysis at early development stages leads to an extensive use of models. The main problem is to validate the models used to predict performance when the software system is not available.
To satisfy the needs exposed in the previous analysis, our research group has focused its work on the development of HeSSE (Aversa et al. 1998a, Aversa et al. 1998b, Mazzocca et al. 2000) (Heterogeneous System Simulator Environment), a simulation environment designed to model complex computer systems, and in the definition of a methodology can be of help in the design of a DHS. Thanks to its component-based approach, it enables simulation model composition, as well as automatic and

semiautomatic model validation and tuning. A graphical Environment helps in building complex models.
The paper is organized as follows: the next section will introduce HeSSE, the model life cycle, its modeling technique, the tools usage. Then we will describe the adoption of the tool on a real complex application, sweep3d (only in the final version of the paper). The paper closes with a large analysis of related works and their comparison with the HeSSE approach.

HESSE

HeSSE is a simulation tool that, using a compositional modeling paradigm, allows the user to simulate the performance behavior of a wide range of distributed systems for a given application, under different computing and network load condition.
The compositional modeling approach allows to easily describe Distributed Heterogeneous Systems that are modeled by interconnecting simple components. Each component reproduces the performance behavior of a section of the complete system at a given level of detail.
A HeSSE component is basically an object, hard-coded with the performance behavior of a section of the whole system. More detailed, each component has to reproduce both the functional and temporal behavior of the subsystem it represents. In HeSSE the functional behavior of a component is the service set that it exports to the other components. So connected components can ask other components for services. The temporal behavior of a component describes the time spent servicing. System modeling is performed primarily at the logical architecture level.
For example, physical-level performance, such as the one resulting from a given processor architecture, is generally modeled with simple analytical models or by integral, and not punctual behavioral simulation. In other words, the use of a processor to execute instructions is modeled as the total time spent in the processor without considering the per-instruction behavior.
Thanks to the chosen approach, HeSSE is capable of describing easily very complex Distributed Heterogeneous Systems at any given level of detail.
HeSSE uses traces to describe applications. A trace is a file that records all the actions of a program relevant for simulation. For example, the trace for a PVM application is a sequence of CPU burst and requests to the run-time environment. Each trace is the representation of a specific execution of the parallel program. Traces can be obtained either by application instrumentation and execution on a host system or by using prototype-oriented software description languages.

SIMULATION DEVELOPMENT CYCLE

A simulation session in HeSSE can be represented graphically as in Fig. 4. It is subdivided in three steps: System Description, Simulation and Results Analysis. It is important to point out that this analysis is just a step in the overall performance analysis methodology.

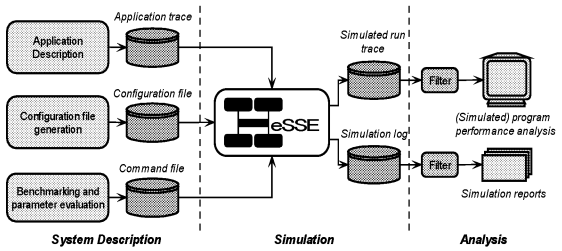


Figure 1. HeSSE Analysis Cycle

The System Description phase includes application execution description, the complete system architecture model definition and evaluation of the time parameters (model tuning). The description of this step will be expanded later.
The application execution description is based on trace file approach: each trace file contains a sequence of the events relative to described task and it is used to drive the simulation engine. The duration of each CPU burst, extracted from the trace file, is processed in order to derive the duration on the final target, thus taking into account the effect of different machine speeds. As mentioned before, task interactions with O.S., run-time supports, I/O devices and networks are instead dealt with by simulation.
Trace files can be customarily obtained through program instrumentation: the program is instrumented and executed on a host system producing trace files. However, the full methodology makes it also possible to adopt a prototype-based approach: the application is described as a prototype, and the trace files generated through prototype interpretation.

HESSEGRAPH

HeSSEGraph is a graphical front-end to the HeSSE simulation engine. It makes possible to create in a user-friendly way configurations of systems to be simulated. It is completely integrated with the simulation engine, and so it is possible to "draw" the hardware and software configuration, to launch the simulation and to visualize the results directly from the graphical environment.

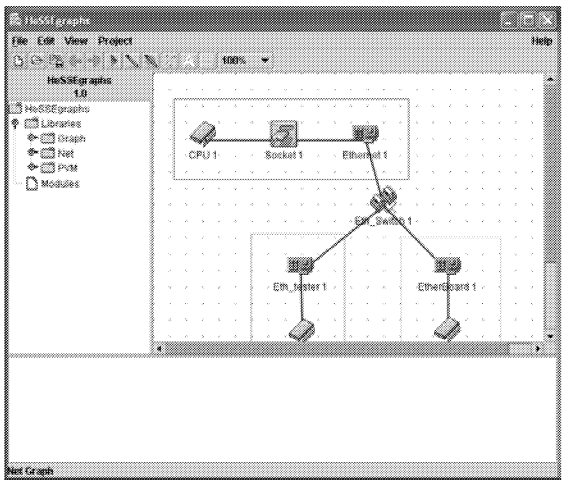


Figure 2. HeSSEgraph Environment

Figure 2 shows a screenshot of HeSSEGraph. On the left side there is the library section of the environment. Here is possible to choose the component needed to "assemble" the configuration. On the right side there is the main window where the current configuration is shown and the new components, added to the configuration, are connected. At the bottom there is the communication area, where HeSSEGraph gives to the user information on the configuration and the simulation execution.

THE MODELING PROCESS

Given a cluster, the modeling process can be carried in three macro steps: *system description*, *model tuning* and *accuracy evaluation and model validation*.
For obtaining the system description, it is firstly possible to use a simple, natural language description. For example, our cluster Cygnus is a five-node cluster in which the front end is a dual Pentium II 350MHz workstation and the other 4 nodes are dual Pentium III 1Ghz machines. The nodes are interconnected via a standard Ethernet switched network. Then it is possible to obtain a high-level description using the HeSSEGraph tool.

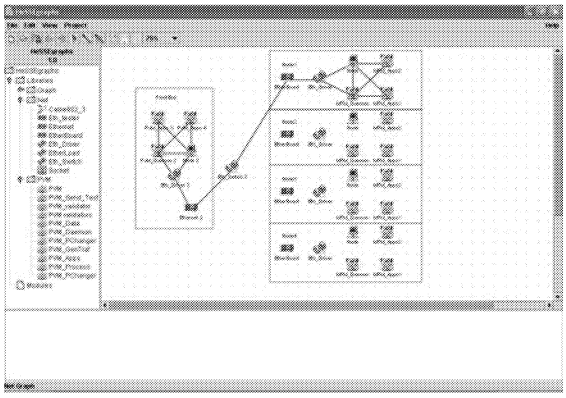


Figure 3. The Cygnus HeSSe Model

As we can see from Fig. 3, in HeSSEGraph the models are plain graphs built by the composition of components connected through arcs.
The construction of the high-level description needs the analysis of the entire cluster structure, hardware and software. The presence of a middleware layer, MPI or PVM, lets the cluster to exhibit the behavior of a single multiprocessor machine. This observation is necessary to obtain a correct high-level description.
The description can be built straightforwardly, using predefined components, but it does not describe the real behavior of the system. In fact, the high-level description describes not a specific cluster, but a class of clusters, because it does not contain any temporal information. The objective of the next step, *Model Tuning*, is to obtain all the parameters that specialize the cluster class to a close description of the given computer system. While high-level description can be obtained by simple natural-language descriptions, using predefined objects, the specializing parameters are harder to obtain. HeSSE offers special tuning components that, using an automatic learning process, allow to compute the optimal parameters for the given system. A detailed description of the tuning process is available in (Mancini et al. 2003).
The full process takes place in the following way: standard benchmark results are collected on the target system, a special HeSSE configuration file is built adding tuning components to the previously defined high-level description, an automatic learning process takes place, and, finally, the simulation error is evaluated validating the model.

APPLICATION DESCRIPTION

The HeSSE models based on Component composition give a description of the system, but contain no detail about the application behaviour, which is described through traces, as previously pointed out.
Traces are usually obtained using code instrumentation, a well-known approach (Aversa et al. 1998). An instrumentation library producing HeSSE trace format is available; it supports Fortran, C, MPI and PVM.
As an alternative to the instrumentation and the tracing of the target application, it is possible to use the MetaPL description language (Mazzocca et al. 2001). This is not a real parallel language, but just a simple and concise notation to support forward and reverse development cycles.
The MetaPL description language is made up of a core, with very limited description capabilities, and language extensions that expand the core language, adding new commands specific to a programming paradigm or to a library or programming language.
The core notation makes it possible to describe at high-level parallel or sequential code, task creation/termination and minimal synchronization constructs.
Being based on XML and XSLT, MetaPL allows describing a parallel or sequential code as a hierarchical structure whose building blocks are different types of code blocks. Each block encapsulates sections of code, and may have attributes such as the expected execution time or the name of a cost function, which gives the (expected) execution time as a function of program inputs.
XSLT filters can be applied to the MetaPL code to obtain HeSSE traces or HTML documentations. New filters can be developed for special operations.

CASE STUDY: SWEEP 3D

The benchmark code SWEEP3D (Adve V. S., et al.. 2000, Kerbyson, D.J., Harper, J.S., Craig, A., Nudd, G.R. 1996) represents the heart of a real ASCI application. It solves a 1-group time-independent discrete ordinates (Sn) 3D cartesian (XYZ) geometry neutron transport problem. The target implementation adopted comes from ASCI Blue Benchmark, and both the code and a complete description of the parallel implementation are available.
The Sweep3D performance analysis process took place following the previously described steps: The target cluster (Cygnus) was modeled using the graphical tool (HeSSEgraph) and tuned through the special components. Being available the complete code, the traces were obtained through code instrumentation.

Predicted application completion times are shown in table 1. It is important to point out that the relative error is always under 10%. The worst case corresponds to the smallest dimension of the problem.

Table 1: Real and Simulated Completion Time

	Real	Simulated	Error (%)
20	313	337	7.67
30	1033	1045	1.16
40	2202	2200	0.09
50	3981	4158	4.45
60	6989	6905	1.2
70	10940	10923	0.16
80	16356	16171	1.13
90	23219	23642	1.82
100	31285	30866	1.34

RELATED WORK

The need for methodologies, techniques and tools for performance oriented development of parallel application led in past years to the development of several interesting tools for performance prediction and analysis.

PARSE (PARallel Software Engineering) (Gorton et al. 1993, Jelly et al 1996a, Jelly et al. 1996b) is a software engineering methodology to facilitate the design of reliable and reusable parallel and distributed (software) systems. The methodology features are:

- a graphical design notation;
- Language- and architecture- independence;
- Formal verification;
- Performance evaluation.

PARSE defines a visual language for the description of the problem composition and solution design in terms of processes and defines a way to construct this description in a simple way. The design can be translated in Petri Nets, used for the analysis of the designed system. The methodology was applied to real problem, such as large Database systems or Client/Server applications.

Even if the methodology is very interesting and innovative, it is lacking as far as the architecture representation is concerned: the system is analyzed at specification level, and the independence from implementation details means that the target architecture and used languages are neglected in the model. This approach can be adopted only in the first design stage, ad Software Performance Engineering (see later) showed that in these cases some problems came out.

C.U. Smith and L.G. Williams coined the "Software Performance Engineering" name when the development of the SPE•ED tool was started and the underlying methodology designed (Smith C.U., Williams L.G. 1997a, Smith C.U., Williams L.G. 1997b). In a way similar to PARSE, this tool defines a visual language to describe the system under design. The graphical description is translated in queuing networks. System usage is modeled with synthetic workloads. The authors found a way to compose complex queue networks. The main differences with PARSE is that the SPE•ED visual models describes both hardware and software components of the system.

The adoption of queuing networks helps in gathering quantitative information in an efficient way. Experience on known systems can be simply reused both to recognize good synthetic workloads and to validate the models. The tools and underlying methodologies were adopted in many real systems and some new experiences are focused on distributed systems, such as CORBA (Jelly, I., Gorton,, I., Croll, P.R.. 1996b).

SPE•ED was a very innovative solution, and introduced many new concepts and ideas. It is the first tool that explicitly regards the system architecture in the model design together with the software. The tool aims to model classical business systems and operational software. Distributed systems are modeled only as natural extension of sequential software systems and programming paradigms (Object Oriented).

The PAMELA (PerformAnce ModEling Language) project (Gemund, A. 1993) aims to the development of a modeling methodology that enables fast development of low complexity, parameterized performance models of parallel programs running on shared-memory as well as distributed memory machines. The methodology is based on the use of Pamela, a process-oriented performance simulation language. The program and the machine are modeled in terms of the Pamela models and compiled to a symbolic performance model. Symbolic Performance models can be easily analyzed to predict the system performance behavior in a very fast way.

The methodology is original and interesting, but leads to very low accuracy for performance prediction (errors higher than 100%). It can be adopted just for the system specification and first design. The target is explicitly process-oriented applications on classical parallel systems.

The POEMS project (Adve V. S., et al.. 2000) aims to create and to demonstrate a capability for prediction of the end-to-end performance of parallel/distributed implementations of large scale adaptive applications. POEMS modeling capability spans applications and operating systems, including parallel I/O, and architecture. Effort focus on the areas where there is little convention wisdom such as execution behaviors of adaptive algorithms on multi-level memory hierarchies and parallel I/O operations.

POEMS provides:

- A language for composing models from component models.
- Derivation of models of applications as data flow graphs from HPF programs,
- A library of component models spanning from workloads to memory hierarchies and at levels of resolution ranging large grain data flow graphs to instruction streams and from probabilistic to fully deterministic.
- parallel execution of the models

- a knowledge base of performance data on commonly used algorithms parameterized for architectural characteristics.

POEMS development is driven by modeling a full-scale LANL ASCI application code executing on an ASCI architecture.

PACE (Kerbyson, D.J., Harper, J.S., Craig, A., Nudd, G.R. 1996) (PerformAnCE analysis Environment) is a toolset for predicting and analyzing performance issues in parallel systems. The overall system is modeled with a three level Model: Application (subdivided in application and subtasks layers) , Parallel template and Hardware.

First level models the software behavior of the system, descriptions are based on a dedicated language (PSL) that characterizes the application in terms of workloads on the underlying system. The second level (Parallel template) evaluates the effects of the parallelization on the workloads. The last level (Hardware) evaluates the workload on the target cluster.

CONCLUSIONS

Both PDSE and SPE focus on the software side in complex systems, usually parallel or distributed, but both ignore problems linked to the underlying hardware architecture. If the definition of clear and effectively applicable rules is difficult in PDSE and SPE, things are decidedly worse for DHS systems, where the target computing environments are flexible, modular and easily re-configurable. Here performance analysis should not only verify if software development choices are compatible with the expected objectives, but could also suggest the use of a larger or differently-configured system.

Software Engineering, and its specialized branches like PDSE and SPE, focuses on software and its life cycle. In DHS systems, the boundary marker between software and hardware is not so clear. Many times it is easier to buy new hardware systems or update existing ones than develop new application versions. This means that the design phase should be faced with methodologies that let the system designer to easily move from hardware to software components.

Approches like PACE and POEMS goes in the same direction of our HeSSE, but focuses more on the application behavior than on the system environment. The plain model adopted by HeSSE and its consequences on the system modeling cannot be found in any other tool.

REFERENCES

Aversa, R., Mazzeo, A., Mazzocca, N., Villano, U. "Heterogeneous System Performance Prediction and Analysis using PS" *IEEE Concurrency* 6 (July-Sept. 1998), pp. 20-29

Adve V. S., et al.. 2000. "POEMS: End-to-End performance Design of arge Parallel Adaptive Computational Systems" *IEEE transactions on Software Engineering* 26, N. 11 (Nov.), pp. 1027-1048.

Gorton, I., Jelly, I., Gray, J. 1993. Parallel Software Engineering with PARSE. *In Proc. IEEE COMPSAC 17th Int Computer Science and Applications Conference*, IEEE Press, Phoenix, USA (1993) 124–130

Jelly, I., Gorton,, I.. 1996a. "The PARSE project." *Software Engineering for Parallel and Distributed Systems*: 271-276

Jelly, I., Gorton,, I., Croll, P.R.. 1996b. "Software Engineering for Parallel and Distributed Systems", In *Proceedings of the First IFIP TC10 International Workshop on Parallel and Distributed Software Engineering*, March 1996.Chapman & Hall 1996

Owens J. 1995, The ASCI Sweep 3d benchmark Code http://www.llnl.gov/asci_benchmarks/asci/limited/swee p3d/asci_sweep3d.html

Smith C.U., Williams L.G. 1997a. "Software Performance Engineering for Object-Oriented Systems", *Proc. CMG*, CMG, Orlando USA (1997)

Smith C.U., Williams L.G. 1997b. "Performance Engineering Evaluation of Object-Oriented Systems with SPEED", *Computer Perfomance Evaluation Modeling Techniques and Tools*, in R. Marie et. Al. (eds.) Lecture Notes in Computer Science, Vol. 1245, Springer-Verlag, Berlin (1997)

Aversa R., Mazzeo A., Mazzocca N., Villano U., 1998a. "Heterogeneous System Performance Prediction and Analysis using PS", *IEEE Concurrency*, Vol. 6, July-Sept. 1998, (pp. 20-29)

Aversa R., Mazzeo A., Mazzocca N., Villano U. 1998b. "Developing Applications for Heterogeneous Computing Environments using Simulation: a Case Study". *Parallel Computing*, Vol. 24 1998, (pp. 741-761)

Mazzocca N., Rak M., Villano U. 2000. "The Transition from a PVM Program Simulator to a Heterogeneous System Simulator: The HeSSE Project". *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, in J. Dongarra et al. (eds.) Lecture Notes in Computer Science, Vol. 1908, Springer-Verlag, Berlin 2000, (pp. 266-273).

Mazzocca N., Rak M., Villano U., 2001. "MetaPL: a Notation System for Parallel Program Description and Performance Analysis" *Parallel Computing Technologies*, in Malyshkin. V. (ed.), Lecture Notes in Computer Science, Vol. 2127, Springer-Verlag, Berlin 2001, (pp. 80-93)

Mancini E., Rak M., Torella R.,Villano U. "Off-line Performance Prediction of Message Passing Application on Cluster System". *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, in J. Dongarra et al. (eds.) Lecture Notes in Computer Science, Vol 2840, Springer-Verlag, Berlin 2003, (pp. 45-54)

Gemund, A. 1993. "Performance Prediction of Parallel Processing Systems: The Pamela Methodology" in *Proc. ACM International Conf. On Supercomputing,* Tokio, ACM, July 1993, pp. 318-327.

Kerbyson, D.J., Harper, J.S., Craig, A., Nudd, G.R. 1996. "PACE: A Toolset to Investigate and Predict Performance in Parallel Systems" *Presented in European Parallel Tools Meeting, ONERA, Paris, October 1996.*

DISTRIBUTED SIMULATION APPLICATIONS

MODEL INTEGRITY AND DISTRIBUTED PROCESS INTERACTION SIMULATION

Hans P.M. Veeke, Jaap A. Ottjes
Faculty of Mechanical Engineering and Marine Technology, OCP
Delft University of Technology
Mekelweg 2, 2628 CD Delft, the Netherlands
E-mail: H.P.M.Veeke@wbmt.tudelft.nl, J.A.Ottjes@wbmt.tudelft.nl

INTRODUCTION

Simulation has become a common tool in research projects and large logistic and manufacturing design projects.
Because of the growing scale, the need for distributed simulation grows rapidly also. Usually it is developed from a general information system approach e.g. by means of HLA (Kuhl et al., 1999] and CORBA [Adamski, Hiller, 1998].
Technically these approaches focus on 'data-integrity', but no provisions are made for model-integrity. Above that, complete knowledge about the final simulation structure is required. The iterative course of a design project however, requires a smooth (and preferably transparent) transition from stand-alone to distributed simulation.
The approach as described in this paper, preserves model integrity for process interaction simulation and is implemented in the free source language TOMAS as can be found on TOMASWEB.COM
First the basic elements and process interaction methods for stand-alone simulation are described and then a short description of the concepts used in TOMAS for distributed simulation are explained. The concept is restricted to conservative distributed simulation.

STAND ALONE SIMULATION

According to Zeigler[2000] the process interaction world view identifies the elements of the model and describes the sequence of actions for each element in a structured way. In the process interaction world view, the simulation is considered a collection of interactions among processes. This world view focuses on a sequence of events and/or activities that are logically connected. There are two different views to connect events and activities. Each view corresponds to a different assumption as to what are the active and passive elements in building a model. In the first view the active elements are taken to be the elements that do the processing, in the second view the active elements are the flow elements. The first one is called pure "process interaction" and the second one the "transaction" world view. From a modeler's point of view events have no explicit meaning. From a processor's point of view the system is still based on event scheduling and an event scheduling mechanism is still required.
The "pure" process interaction world view has been implemented in an object oriented way in the simulation language TOMAS [Veeke, Ottjes, 2000]. The approach matches with a formal systems approach to model logistic and manufacturing systems [Veeke, 2003]. In order to connect them in an interdisciplinary way to behavior descriptions of these systems, an intermediate Process Description Language PDL has been developed [Ottjes, Veeke, 2002].

TOMAS implements two basic object classes for stand alone simulation: TomasElement and TomasQueue. The TomasQueue class supports fast and easy methods to collect, select, sort and register TomasElements during the course of a simulation run. Table 1 shows the basic methods of the TomasQueue class.

Method	Description
AddToTail(Elm) AddToHead(Elm)	To add element Elm to the tail or head of the queue
AddSorted(Elm,Sort)	To add element Elm to the queue sorted by procedure Sort
PutBefore(E1, E2) PutAfter(E1,E2)	To put E1 before or after E2 into the queue
FirstElement LastElement	To retrieve the first or last element in the queue
Successor(Elm) Predecessor(Elm)	To retrieve the successor or predecessor of Elm
Length MeanLength MinLength MaxLength	To retrieve the actual, mean, minimum or maximum length of the queue
MeanWT MinWT MaxWT	To retrieve the mean, minimum or maximum length of stay

Table 1. Basic methods of TomasQueue

With the TomasElement class both the active and passive model elements can be represented. Passive TomasElements are the model elements being handled or transformed by the active TomasElements. The latter have their own Process description method. All TomasElements have an Identity attribute, an Arrival Time, an Event Time and a Status.
The Status attribute is directly related to the process interaction approach. In TOMAS, 4 values for the Status are defined (see fig.1.)

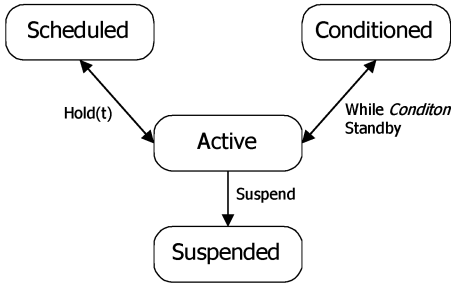


Fig. 1. State values of a Process

A process status can be:

- *Suspended*: the TomasElement "sleeps" and no future event has been defined.
- *Active*: the TomasElement is the "current" element and executing it's event.
- *Scheduled*: the TomasElement is waiting for the simulation clock to arrive at a predefined event time to become "active".
- *Conditioned*: the TomasElement is waiting for a condition to become false and then to be come "active".

As shown in figure 1, a TomasElement must be in the state "Active" to be able to change its state. The sequencing mechanism of TOMAS offers three methods for this purpose:

- Hold(t): The execution of the process description is interrupted and will be resumed at the current simulation clock time + t.
- While *Condition* Standby: the execution of the process description is interrupted and will be suspended as long as *Condition* holds.
- Suspend: the execution of the process description is interrupted indefinitely.

All states except the "Active" state, represent time periods; the Active state represents an event.

The 'Suspended' state is the initial state of every TomasElement. Figure 1 shows that an "external" cause is required to change the process to another state. Here we reach the important issue of "process interaction".

STAND ALONE PROCESS INTERACTION

The only way for a TomasElement to become the active element is to become Scheduled by means of another TomasElement. If the simulation clock time reaches the moment where the element is Scheduled, the sequencing mechanism causes it to become Active. The process description uses the above mentioned method to change to the other states autonomously during the course of its process. However, there can be circumstances where the normal process course – and thus the states Scheduled, Conditioned and Suspended, should be interrupted e.g. because of a machine disturbance, or the arrival of a new order etc. For these situations the TomasElement class offers a number of methods to intervene with the processes of other elements. They are shown in figure 2.

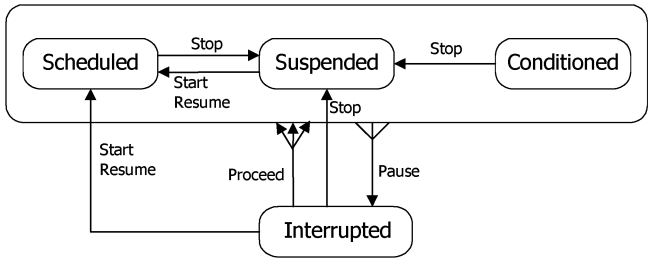


Fig. 2. Process interactions

In the upper part of figure 2 the three time consuming states of figure 1 are shown. For example, a TomasElement can force another element to change from the state Suspended to Scheduled by means of the Start or Resume method. A short description of each method is given in the table below.

Method	Description
Start(t)	Start the process of the element at simulation time t, beginning with the first statement of description.
Resume	Resume the process of the element, with the statement immediately following the last executed statement, which can be Hold, Suspend or StandBy
Stop	Make the process of the element Suspended
Pause	Pause the execution of the process, but remember the state it is in and the time left for its next event
Proceed	Resume the execution of the process as remembered by the Pause method

Table 2. Process interaction methods

The Pause method introduces a new process state: "Interrupted". Interrupted can be compared with Suspended with the difference that the state of the process is completely remembered. From the Interrupted state one may decide to continue the process by means of Proceed, to start it from the beginning by means of Start or to Resume the process, skipping the state it was in.

Figure 2 shows clearly that the methods can only be invoked in certain situations, depending on the state of the process. Starting a process with Start(t) is only allowed if the process is in a Suspended or Interrupted state. The sequencing mechanism of TOMAS preserves this type of *"process integrity"*.

DISTRIBUTED SIMULATION

In research situations and large industrial projects, it often happens that a simulation model grows in size and detail and more than one modeler is constructing separate parts of the model. In these cases it will be very profitable if the model can be split up easily into different stand-alone models that will be synchronized to the same simulation clock. Until now the client-server concept of TOMAS facilitates distributed simulation for both a fast conservative approach and real-time situations. However, the transition from a stand-alone model to a set of distributed models was quite labor-intensive.
The conservative approach can best be described by distributing the processes over several processors, but allowing only one processor to be really 'processing' at any one moment. It is just as "quasi-parallel" as the stand-alone situation. However, instead of one sequencing mechanism, there are several sequencing mechanisms now. Because of this (and of course because of distributing "data") the integrity of the model is not automatically preserved. To find a solution it is necessary to define the term integrity precisely.
Model integrity is defined as the preservation of correct values and correct process states at any moment at any model during a simulation run. The preservation of correct values in a stand-alone situation is automatically preserved by TOMAS, because it is a single processor application where no threads or agents are being used. There is also only one sequencing mechanism present, which takes care of the process states. From the above definition it becomes clear that we should distinguish data-integrity and process-integrity.

DATA-INTEGRITY

Data-integrity could be achieved by using HLA or CORBA for example. However, the application of HLA and CORBA is not straightforward and again very labor-intensive. TOMAS uses a simple, but fast TCP/IP-based message concept that can be made transparent for the modeler, where the data-integrity of TomasElements and TomasQueues is concerned.

The basic assumptions to achieve this type of integrity are:

- Only one processor is actually processing
- Only one process is actually 'Active' during the simulation run
- The "same" TomasElement may be present in different models, but not necessarily with the same attributes.

The last assumption needs some explanation. Usually data-integrity leads to the requirement that each object may only be present once and only once in the whole model structure. There are two reasons to drop this requirement:

- in each participating model, the same object can play a different role with different attributes. For example, a container in a transportation model only needs a size and weight, while the same container needs other attributes like "empty/not empty" and "reefer/non-reefer" in a storage model.
- By restricting this approach to the conservative distributed concept, only one model is running at any moment. So there is only one source, which can alter the attribute values.

Data-integrity can now be preserved by updating the common and generally defined attribute values at the moment that the active model returns control to the server in order to advance the simulation clock. The implementation of data-integrity can thus be achieved by defining one object (class) definition in one model to be the "home element" and the other ones in the other models to be "remote elements". It is up to the modeler to decide on which class definition is to be considered the home definition. As soon as a model containing remote elements returns control to the server, the sequencing mechanism communicates all changed objects to the server. The server distributes these updates to the "home" models involved. To realize this distinction TOMAS now recognizes a "TRemoteElement" class. The home element can still be defined as a TomasElement class.

Example

Suppose a distributed model of container transfer contains a.o three submodels: a generator model, an

AGV transportation model (Automatic Guide Vehicles) and a Storage model. The modeler decides to define the container class in the generator model as the home element class. The class definitions in the model could look like the ones shown in table 3 (in Delphi Pascal).

Generator model
 	
Tcontainer = class(TomasElement)
  Size: Integer;
  Weight: Integer;
  Reefer: Boolean;
  Empty: Boolean;
  Stored: Boolean;
End;

AGV Transportation model
 	
TContainer = class(TRemoteElement)
  Size: Integer;
  Weight: Integer;
  OnAGV: AGVReference;
End;

Storage Model
 	
TContainer = class(TRemoteElement)
  Size: Integer;
  Reefer: Boolean;
  Empty: Boolean;
  StackPosition: XYZ_Position;
  Stored: Boolean.
End;

Table 3. Home and remote elements

Size, Weight, Reefer and Empty are considered the common attributes here. At the first time after receiving control from the server, a model refers to a container, the attribute values of the home element are "automatically" communicated through the distributed sequencing mechanism of TOMAS. If other models need to know the StackPosition, the modeler is free to add this attribute to the definition in the Generator model. If the container is stored in the Storage model, the "Stored" attribute is set to TRUE and when this model returns control to the Server again, this attribute value is automatically updated by the distributed sequencing mechanism.

PROCESS-INTEGRITY

Process-integrity (as defined by figure 2) should be preserved by and between the different sequencing mechanisms. In this way it can be made (almost) transparent to the modeler.
Two situations should be distinguished:

1. the process description of an element is completely contained in only one participating model.
2. The process description is split into parts, where each part is contained in a different model.

In situation 1 the process interaction methods of figure 2 can be called from any model. Suppose the home element is defined in the model containing its process description. The other models contain the remote element definition, but remote elements also have process interaction methods. In this case, a call to "Start" the process of a remote element is interpreted by the distributed sequencing mechanism and translated into a "Start message" for the home element. The sequencing mechanism in the home model receives the message and performs the required action if it is allowed (just as it did in the stand alone case).
Situation 2 however is more complicated. The distinction between home and remote element definitions is not sufficient to guarantee process integrity. It should be guaranteed that the element can be active, scheduled or conditioned in only one model at a time. To achieve this an extra process state "Disabled" is introduced. At the moment of creation the home element initializes to the state "Suspended". Above that, all remote elements are considered "Disabled" initially. Process interaction statements are considered to apply to the one element which is not "Disabled". Now the method to perform the state transitions to and from the state "Disabled" must be defined. For this purpose the element method "SwitchToModel(*modelname*)" is introduced. The method is defined for both home and remote elements.
First of all this method can be called by the element itself whenever it is in the Active state (see figure 3).

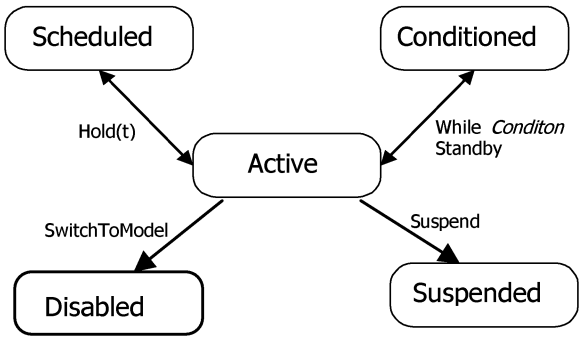


Figure 3. State values of a distributed process

By calling "SwitchToModel" the element itself will become "Disabled" and the distributed sequencing mechanism sends a message to the server to enable the corresponding (home or remote) element in the model mentioned. If the sequencing mechanism in

this model receives this “enable”-message it changes the state of the element concerned to “Scheduled” at the current simulation clock time. The process of this element will resume its course from the statement following a SwitchToModel or –if it is the first time of enabling- will start its process description.

Secondly, if the SwitchToModel is called when the element is not Active, the process is required to be in a Suspended state. It is physically impossible to be scheduled, conditioned or interrupted in two models at a time. Therefore SwitchToModel will only be accepted by the sequencing mechanism if the element is Suspended or Active. As described above, when the element is “enabled” it changes from a Disabled to a Scheduled state. Figure 4 now shows the process interactions in a distributed environment.

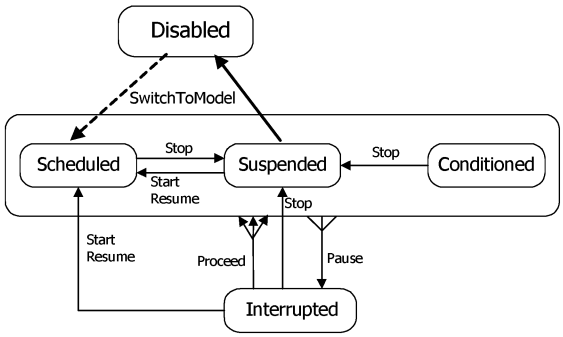


Figure 4. Distributed Process Interactions.

Example

Suppose a transportation system with Automatic Guided Vehicles (AGV) provides the transport of containers between quay cranes and stacking cranes. Arriving at a quay crane AGV’s line up in a QuaycraneQueue and drive forward until they reach the transfer position. There an AGV is loaded and drives to a stacking crane. At a stacking crane it searches a free position and waits until the container is picked. Then the AGV drives to the next quay crane.
Initially the process of an AGV was completely described in one stand alone model. The process description of an AGV looked like below (assuming an AGV starts at a transfer position near a stacking crane).

```
Process of an AGV

        While TRUE Do
        Begin
          While JobList.Length = 0 Do
            StandBy;
          Hold(Drivetime);
          EnterQueue(QuaycraneQueue);
          Suspend;
          Hold(DriveTime) ;
          EnterQueue(TransferPositions) ;
          Suspend;
        End;
```

The model is now split up in three different models: a Quay crane Model, an AGV model and a Stacking model. In order to study the behavior and control at quay cranes and stacking cranes further, the process of an AGV is also split up. This is now easily accomplished as shown below.

```
AGV_model

Process of an AGV
        While TRUE Do
        Begin
          While JobList.Length = 0 Do
            StandBy;
          Hold(Drivetime);
          SwitchToModel(Quaycrane_model);
          Hold(DriveTime) ;
          SwitchToModel(StackingModel)
        End;

Quaycrane_model

Process of an AGV
        While TRUE Do
        Begin
          EnterQueue(QuaycraneQueue);
          Suspend;
          SwitchToModel(AGV_model) ;
        End;


Stacking_model

Process of an AGV
        While TRUE Do
        Begin
          EnterQueue(TransferPositions);
          Suspend;
          SwitchToModel(AGV_model) ;
        End;
```

Now in each of the models the descriptions can be further detailed independently and concurrently.

CONCLUSIONS AND FUTURE RESEARCH

It has been shown that the use of distributed simulation can be logically connected to a stand alone approach. In this article distributed simulation was restricted to the conservative (i.e. quasi-parallel) concept. In this case the requirements of model integrity should be preserved as they already are in the stand alone environment. In TOMAS this has been implemented by distinguishing home and remote elements, by introducing an extra process

state “Disabled” and by adding an extra process interaction method “SwitchToModel”. In the near future the concept will be expanded for real-time distributed simulation in order to support prototyping of real equipment and control software.

The model integrity concept will be used in a starting research project VISITOR, which aims to construct a virtual industrial system for education and research purposes.

REFERENCES

Zeigler, B.P., Praehofer, H., Kim, T.G., 2000, “*Theory of Modeling and Simulation*”, Academic Press, San Diego, 2nd Edition.

Ottjes, J.A., Veeke, H.P.M., 2002, “*Prototyping inProcess Oriented Modeling And Simulation*”, Proceedings of the 16th European Simulation Multiconference ESM 2002, pp. 20-26, Darmstadt, Germany

Kuhl,F., Dahmann,J. Weatherly,R.,1999*, ”Creating Computer Simulation Systems: An Introduction to the High Level Architecture”,* Prentice Hall, ISBN 0130225118

Veeke, H.P.M., Ottjes, J.A., 2000, “*TOMAS: Tool for Object oriented Modeling And Simulation*”,Proceedings of Advanced Simulation Technology ASTC 2000, pp. 76-81, WashingtonDC, SCS International

Adamski, D., Hiller, M., 1998, “*CORBA in Simulation Tasks*”, Proceedings of EUROSIM 1998, Helsinki, Finland

Veeke, H.P.M., 20003, “*Simulation Integrated Design for Logistics*”, Dissertation Delft University of Technology, ISBN 90-407-2417-2

TRANSPARENT DISTRIBUTED DISCRETE EVENT MODELING

Jaap A. Ottjes
Delft University of Technology, Sub Faculty of Mechanical Engineering and Marine Technology
Transport and Logistic Technology Group
Mekelweg 2, 2628 CD Delft
The Netherlands
J.A.Ottjes@wbmt.tudelft.nl

Hans P.M. Veeke
Delft University of Technology, Sub Faculty of Mechanical Engineering and Marine Technology,
Industrial Engineering Group
Mekelweg 2, 2628 CD Delft
The Netherlands
H.P.M.Veeke@wbmt.tudelft.nl

ABSTRACT

Stand-alone modeling and distributed modeling show considerable differences in methodology and amount of modeling effort needed. An approach is discussed to design and develop distributed models with the ease of stand-alone modeling and without the overhead usually associated with distributed modeling. The main objective is to obtain interchangeability of a stand-alone model and its distributed equivalent, maintaining the model structure in all models and to minimize code alteration and overhead. Important requirements are reproducibility of simulation experiments and model integrity. The process interaction modeling approach appears to be very appropriate for this. Most attention is paid to the interaction between model elements in stand-alone as well as distributed models. Some examples are show in pseudo language and demands on simulation software are derived.

INTRODUCTION

In traditional applied modeling and simulation of discrete systems generally 'stand-alone' models are used. Several languages and packages are available for this. A stand-alone discrete event simulation model can be characterized by having one time sequencing mechanism. Usually the model is contained in one program. Distributed discrete event simulation models are constituted of interacting sub models or member models. These sub models may reside in one or more computers. Sub models may have their own local time sequencing mechanism but have to be mutually synchronized by a central clock.

Reasons to use distributed modeling instead of or in addition to stand-alone modeling may be:

- Enabling concurrent modeling by several modelers each developing a part of the total model.
- Enabling zooming into a part of the model i.e. to expand one sub model into more detail letting the other models unchanged.
- Running sub models at different locations possibly with human interfaces, for example for training purposes.
- Running the model real time as a virtual environment for testing real resources and control functions, (Ottjes and Hoogendoorn, 1996).

In some cases having an (initial) stand-alone model as well may have advantages in terms of development, speed and testing, (Duinkerken et al, 2002).
In the next chapters the characteristics of process interaction modeling will be discussed in some detail. Then it will be applied to both stand-alone modeling and distributed modeling. Some simple examples will be shown as an illustration and the key demands for transparent modeling will be derived from it.

DISCRETE EVENT MODELING

Zeigler (Zeigler 2000) distinguishes three 'world views' on discrete event systems:

Event scheduling: An event-scheduling algorithm employs a list of future events ordered by increasing event times.
Activity scanning: concentrating on the activities in a system and the conditions that allow an activity to start.
Process interaction being a combination of event scheduling and activity scanning. It can be characterized as identifying the systems elements and describing the sequence of actions of each one. The sequence of actions of an element is called its process. There are two different views corresponding to a different assumption of what are the active and passive elements. The first view is the pure "process interaction". Here the active elements are taken to be the elements that do the processing. In the second view called the "transaction" view the active elements are the flow elements.

Though the transaction view is the most frequently used technique, being able to also describing the processes of the active permanent elements as well increases the flexibility of modeling, (Fishman, 2001). The first language applying this approach is "Simula" (Birthwistle, 1973); two recent tools are Silk (Healy and Kilgore, 1997) and Tomas (Veeke and Ottjes, 2000), (Veeke, 2003).

PROCESS INTERACTION MODELING

The process interaction modeling is quite straightforward using the following recipe:

- Distinguish the relevant classes of elements and their relevant attributes; elements of one class own the same set of attributes.
- Distinguish between active and passive element classes. At this stage the worldview emerges. Provide the process description of the active classes. A process of an active class displays the activities of elements of that class as a function of time. Consequently in a process 'time consuming' commands appear. Further an active element may interact with other elements. Therefore control commands are needed.

TIME CONSUMING COMMANDS

In a process description of an element class so called 'time consuming' commands occur. Time consuming refers to system time during simulation. If a time consuming command appears in a process description the process halts for a certain time interval and later on continues according table 1.

Table 1. Time consuming commands and rules for process continuation.

Command	Continuation rules
Advance	if triggered by "resume" command from other element
Advance τ	automatically after τ time units
Advance while/until condition	automatically after condition=false/true

If more elements are following their process at the same time quasi parallelism is obtained. For that a mechanism is needed that schedules the right processes at the right times in the right order. Moreover it has to bring up to date the system time. This mechanism is called the time sequencing mechanism. Every event based simulation language is provided with such a mechanism.

CONTROL COMMANDS

Active elements need the possibility to act upon the process of other elements in the model influencing the status of processes of these elements.

The basic commands for interaction are listed in table 2. The so-called 'dot notation' familiar in object oriented modeling, is used. Apart from Element classes a Set class is appropriate in modeling. A Set may contain elements. A set is used to represent a physical waiting queue of elements but is also very useful in modeling control and selection logic.

Table 2. Basic commands for interactions between processes of active elements E1 and E2

Command in process of element E1	Result
E2.Start T	Start the process of E2 from its beginning at time T
E2.Interrupt	Immediate interrupt the process of element E2
E2.Resume T	Let element E2 continue at time T with its process (after having been interrupted)
E2.Stop	Stop the process of element E2 immediately

Some typical Set related commands and attributes are listed in table 3.

Table 3. Set related commands and attributes

First	Refers to the first element of the Set
Last	Refers to the last element of the Set
Successor	Refers to the successor of some element
Predecessor	Refers to the Predecessor of some element
Leave / Remove	Methods to remove element from the Set
Enter / Add	Methods to add elements to the Set
Length	Actual number of elements in the Set

STAND ALONE VS. DISTRIBUTED MODELING

It is the purpose of this work to define an approach to develop models with the ease of stand-alone modeling and without the overhead usually coupled with distributed modeling which is the case for example in HLA applications (Fujimoto, 2000), (Klein and Strassburger, 1998). Both distributed model and its stand-alone version should be equivalent. Moreover we want to be able to switch between both stand-alone model and distributed model at any stage of development without code alteration or additional programming.

In a process interaction model the interactions between elements are modeled using the commands in table 1 and table 2 and the set operations are modeled using the commands in table 3. As a consequence elements need access to other elements they have to interact with. In a stand-alone model this is an obvious matter. In a distributed model special facilities are required. Moreover additional requirements are to maintain data and process integrity and reproducibility of simulation runs. In the next examples the stand-alone and equivalent distributed modeling will be illustrated in terms of an informal process description (pseudo)

language (Ottjes and Veeke, 2002) and consequences for the modeling and simulation software are discussed.

EXAMPLE

Suppose we have a generator of loads that have to be transported from a pickup location to various destinations. Automated guided vehicles (AGVs) are carrying out the transportation of the loads. Every load owns its specific travel time to its final destination. This time is an attribute of the class Load.

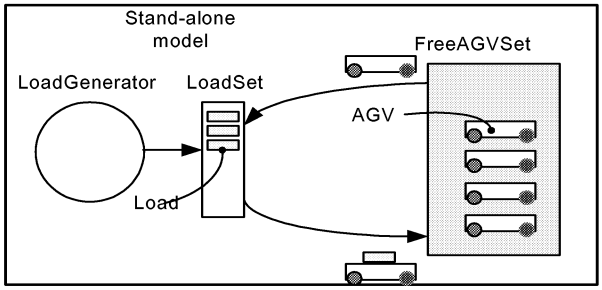


Figure 1. The stand-alone model (model 1)

First we will elaborate the stand-alone model (model 1) of this system. After that a distributed model (model 2) will be derived consisting of two sub models: The *Generator model* containing the Load-Generator and the *Transport model* containing the AGVs. Next a distributed model formed by the same Generator model, a reduced Transport model and for each AGV a separate distributed AGV model (model 3) will be discussed. Finally the AGV model is further elaborated into a control part and a part that represents the physical driving, (model 4). In figure 1 the stand-alone model is depicted.

Model 1: The Stand-Alone Model

Following the modeling rules as given earlier we distinguish two active element classes: the Loadgenerator and the AGV and one passive class: the Load, each with its specific attributes and methods: Figure 2a.

Classes	Attributes
LoadGenerator: ElementClass	ArrivalTime-Distribution TravelTime-Distribution PROCESS
Load: ElementClass	TravelTime
AGV: ElementClass	PROCESS
LoadSet: SetClass	
FreeAGVSet: SetClass	

Figure 2a: Definition Section of the stand-alone model

Two element classes own a process: The Generator shown in figure 2 b and the AGV shown in figure 2c.

("Now" means the current system time; "//" precedes a comment)

```
NewLoad: Load   //local reference
FreeAGV:AGV    //local reference
Loop
  Advance InterArrivalTimeDistribution.Sample
   NewLoad = Load.Create
   NewLoad.TravelTime=   TravelTimeDistribution.Sample
   NewLoad.enter(LoadSet)
   If FreeAGVSet.Length>0 then
     FreeAGV = FreeAGVSet.FirstElement
     FreeAGVSet.Remove(FreeAGV)
     FreeAGV.Resume Now
```

Figure 2b: The LoadGenerator Process in the stand-alone model

```
MyLoad : Load          //local reference
Loop
  While LoadSet.Length > 0 do
    MyLoad = LoadSet.First
     Loadset.Remove(MyLoad)
     Advance MyLoad.TravelTime
     MyLoad.Destroy
  Enter(FreeAGVSet)
  Advance
```

Figure 2c: AGV Process in the stand-alone model

Figure 2d depicts the initialization part of the model.

```
LoadSet            = Set.Create
TheGenerator       = Generator.Create
TheGenerator.Start Now
FreeAGVSet         = Set.Create
Loop(NumberOfAGVs)    //create the AGVs
  NewAGV = AGV.Create
  NewAGV.Start Now
```

Figure 2d: Initialization Section of the stand-alone model

Model 2: Two Sub-Models

Now we will split the stand-alone model into two interacting sub models forming a distributed model:

- o The Generator model containing the classes LoadGenerator and Load and the LoadSet
- o The Transport model containing the class AGV and the FreeAGVSet.

We intend to use the same process code and will discuss the consequences of that. Therefore it is necessary to identify at what occasion one model needs information from or wants to interact with the other model. Looking at the processes the Generator has to 'know' about the FreeAGVSet and AGVs defined in the Transport model and each AGV has to 'know' the

LoadSet and Loads in the Generator model. Moreover an AGV should be able to access the attribute TravelTime of its Load. Further the Generator should be able to remove an AGV from the FreeAGVSet and resume its process. Its sounds reasonable to require that it should be known in a model whether references to a Set or an Element concern 'own' items or items that are created in another sub-model. We will refer to sets and elements in another model as 'remote' set class and 'remote element class'.

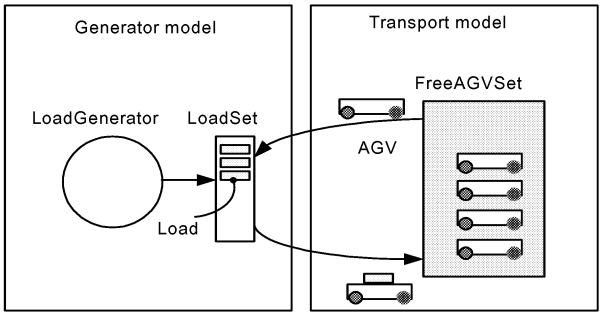


Figure 3. Two sub-models (model 2)

The Generator sub-model then shows as follows:

Classes	*Attributes*
LoadGenerator: ElementClass	ArrivalTime-Distribution TravelTime-Distribution PROCESS
Load: ElementClass	TravelTime
AGV: **RemoteElementClass**	
LoadSet: SetClass	
FreeAGVSet: **RemoteSetClass**	

Figure 4a: Definition section of the Generator model

In the definition section only the definitions of reference to remote element classes and sets are changed accordingly.

LoadSet = Set.Create
TheGenerator = Generator.Create
TheGenerator.Start Now
FreeAGVSet = RemoteSet.Create

Figure 4b: Initialization Section of Generator model:

The initialization section of the Generator model only contains the relevant definitions of figure 2d
The process description in the LoadGenerator class remains unchanged.

The Transport sub- model of model 2 is defined in figures 4a and 4b.
The process description of the AGV class remains unchanged. It is clear that splitting models into sub-models bares the risk of disruption data integrity. For example if in one sub-model an attribute of an element is changed then this alteration should be unambiguous. This problem can be solved in principle by letting each element have an unambiguous owner sub-model. In model 2 for example the owner model of a load is the Generator model.

Every alteration of attributes and status of the element has to be adjusted in the owner model at the right moment. This puts further demands on the implementation of the simulation software. When switching between stand-alone model and distributed model the results of simulation runs should be identical and reproducible. Reproducibility is obtained if the time sequencing mechanism of the distributed model is implemented conservative.

Element: Class	*attributes*
Load: RemoteElementClass	
AGV: ElementClass	PROCESS
LoadSet : RemoteSetClass	
FreeAGVSet: SetClass	

Figure 4a: Definition Section of the Transport model

LoadSet = RemoteSet.Create
FreeAGVSet = Set.Create
Loop(NumberOfAGVs)
NewAGV = AGV.Create
NewAGV.Start

Figure 4b: Initialization Section of Transport-model

Requirements Derived
If we require that the implementation of both stand-alone and distributed models do not require more adapting than in the pseudo model i.e. just discriminate between 'own' and 'remote' element classes and sets and leaving processes unchanged, this determines the requirements on the simulation software so far:

In a distributed model:
- Each sub-model should be able to refer to elements and sets in all other sub models.
- Each sub-model should have access to (read and write) attributes of elements and sets in all other sub-models
- Each sub-model should be able to control the process of elements in all other sub-models (cancel, resume, remove, add, etc.)
- Data integrity should be maintained.

Next the example will be further elaborated into more detail.

Model 3: A Sub-Model For Each AGV
We will now create for each AGV its own AGV model. Consequently the Transport model only contains the FreeAGVSet.

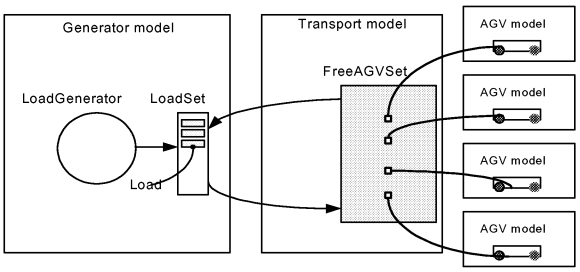


Figure 5: All AGVs have their own sub-model (model 3)

The Generator model as defined in model 2 does not need any adjustment.

The definition section of the Transport model turns to the one shown in figure 6a:

Element: Class	*attributes*
LoadSet : RemoteSetClass	
FreeAGVSet: SetClass	

Figure 6a: Definition Section of the Transport model

And the initialization section is reduced to figure 6b:

LoadSet	= RemoteSet.Create
FreeAGVSet	= Set.Create

Figure 6b: Initialization Section of Transport model

The AGV model

Element: Class	*attributes*
Load: RemoteElementClass	
AGV: ElementClass	PROCESS
LoadSet : RemoteSetClass	
FreeAGVSet: RemoteSetClass	

Figure 6c: Definition Section of the single AGV model

FreeAGVSet	= RemoteSet.Create
LoadSet	= RemoteSet.Create
NewAGV	= AGV.Create
NewAGV.Start	

Figure 6d: Initialization Section of the single AGV model

The process description of the AGV class remains unchanged, see figure 2c.

As a conclusion we can say that going from model 2 to model 3 no statement has changed. They only are re-distributed over the sub-models.

Model 4: Zooming Into The AGV Model

If we now concentrate on the AGV model and especially on the AGV process we see that the process model has three different functions:

1: Waiting for a new Load
2: The selection of a new load; selecting means here simply taking the first load in the LoadSet. Selection however also may consist of any other selection procedure. We can say that the AGV has its own "local intelligence" aboard with respect to selecting loads. Of course the selection of loads also could have been modeled using a central dispatcher located in the Transport model.
3: The actual driving, modeled simply by suspending the AGV process during its loads travel time, see figure 2c.

Zooming

The next step is zooming into the driving part of the AGV process, taking into account a road map, traffic control (Duinkerken et al, 1999), and AGV physical driving characteristics. The information needed from the load should be given in terms of destination rather than travel time. With such a model load assignment algorithms and traffic control methods can be tested and optimized. As simulation speed of distributed models decreases by delays caused by messaging between the sub models, this testing of control algorithms is best be done using the stand-alone model variant.

Splitting Up Processes

A further step is to split up the AGV process in parts implemented in *different* sub-models. In that case an AGV has to be "transferred" from one sub-model to another following in each of these sub-model its process part. Here for example the 'waiting for a load' part of the model could be implemented in the generator model leaving only the 'driving part' in the AGV model. Apart from data integrity now we also have to preserve "process integrity" to be defined as the demand that, if a process of an element is distributed over several sub-models, the element may only be active in one of the sub-models at the time. In all other sub-models its process should be disabled. We will not further go into this matter in this paper. In (Veeke and Ottjes, 2003) the implementation consequences are elaborated.

The final step is the introduction of the real equipment e.g. one or more AGVs. The distributed models then act as a virtual environment in which real AGVs can function in cooperation with simulated ones. Obviously the model should run real time in that case, (Lindeijer, 2003). Figure 7 shows a scale model of an AGV equipped with a computer, a wireless network connection and an interface between the physical driving system and the control part of the AGV process.

It is possible now to control the AGV using a distributed simulation model. The AGV is used in current research.

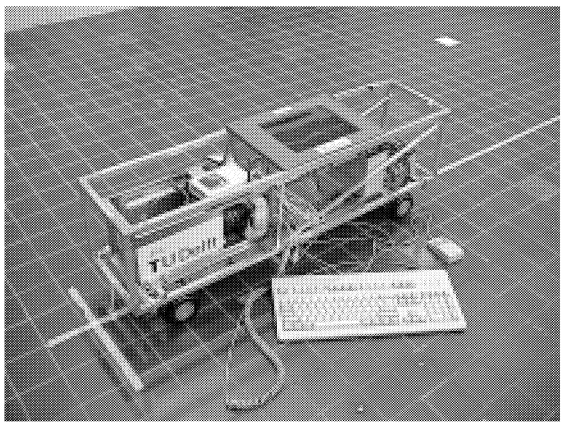


Figure 7. Scaled AGV with a "sub-model" on board contained in a distributed model with wireless communication via the Internet.

CONCLUSIONS

An approach is discussed to design and develop distributed models with the ease of stand-alone modeling. The main objective is to obtain interchangeability of a stand-alone model and its distributed equivalent, maintaining the model structure in all models and to minimize code alteration and overhead. To that end simulation software has to fulfill the following demands:

- Each sub-model should be able to refer to elements and sets in all other sub models.
- Each sub-model should have access to (read and write) attributes of elements and sets in all other sub-models
- Each sub-model should be able to control the process of elements in all other sub-models (cancel, resume, remove, add, etc.)
- Data integrity should be maintained.
- Process integrity should be maintained.
- Simulation runs in both stand-alone model and distributed model should be reproducible.

CURRENT AND FUTURE RESEARCH

The philosophy explained is being implemented in the simulation tool Tomas. The results will be applied in prototyping of logistic systems in production and transportation. Currently a project is carried out to realize "hardware in the simulation loop".

REFERENCES

Birtwistle, G. M., O. J. Dahl, B. Myhrhang, K. Mygrard (1973), Simula Begin, Van Nostrand Reinhold, New York.

Duinkerken, M.B., Evers, J.J.M., Ottjes, J.A. 1999. *TRACES: Traffic Control Engineering System.* Proceedings 31st Summer Computer Simulation Conference. Chicago [SCS] 1999, pp. 461-465.kl

Duinkerken, M.B, J.A. Ottjes, G. Lodewijks, 2002; *The Application of Distributed Simulation in TOMAS: Redesigning a Complex Transportation Model.* Proceedings of the 2002 Winter Simulation Conference (WSC 2002). December 2002. San Diego. ISBN 0-7803-7615-3

Fishman, G.S. 2001. *Discrete Event Simulation. Modeling, Programming, and Analysis.* @001 Springer-Verlag New York, Inc. ISBN: 0-387-95160-1, 52-59

Fujimoto R.M. 2000. *Parallel and Distributed Simulation Systems.* Wiley Series on Parallel and Distributed Computing. Wiley, New York.

Healy, J. R.A. Kilgore, 1997. "Silk,: A Java-Based Process Simulation Language". *Proceedings of the 1997 Winter Simulation Conference, IEEE,*

Klein, U., Strassburger, S., Beikirch, J. *Distributed Simulation with JavaGPSS based on the High Level Architecture.* International Conference on Web-based Modeling and Simulation, Jan. 11.-14. 1998, San Diego.

Lindeijer, D.G, 2003, *"Controlling Automated Traffic Agents"*, Dissertation, Univ. of Techn. Delft, Februari 2003, ISBN 90-5166-948-8

Ottjes, J.A., F.P.A. Hogedoorn, 1996; "Design and control of multi-AGV systems: reuse of simulation software." *Proceedings of 8th European simulation symposium.*
Society for Computer Simulation International Genoa 1996, p. 461-465. ISBN: 1-56555-099-4

Ottjes J. A. , Veeke, H. P.M., , 2002 *"Prototyping In Process Oriented Modeling And Simulation"*,
Proceedings 16th European Simulation Multiconference ESM 2002, pp 20-26, ISBN 90-77039-07-04, Darmstadt, Germany

Veeke, Hans P.M., Jaap A. Ottjes, 2000. Tomas: Tool for Object-oriented Modeling And Simulation. *In proceedings of Advanced Simulation Technology Conference (ASTC2000).* April 16-20, 2000, Washington, D.C. pp. 76-81. The Society for Computer Simulation International (SCS), ISBN: 1-56555-199-0

Veeke, Hans P.M., Jaap A. Ottjes, 2002. TomasWeb: web site: www.tomasweb.com

Veeke, H.P.M, 2003, *"Simulation Integrated Design for Logistics"*, Dissertation, Univ. of Techn. Delft, June 2003, ISBN 90-407-2417-2

Veeke, H.P.M., Ottjes, J.A., 2003, *"Model Integrity and Process Interaction Simulation"*, Submitted to the European Simulation and Modelling Conference (ESM2003) Naples, Italy,

Zeigler B.P., Praehofer H and Kim T.G., 2000. *"Theory of Modeling and Simulation* 2nd Ed. Academic Press, San Diego.

GRAPHICS
VISUALIZATION
SIMULATION

ADVANCED ACCIDENT FLIGHT PATH SIMULATION AND INNOVATIVE VISUAL ANIMATION

Domenico P. Coiro, Agostino De Marco
University of Naples "Federico II"
Dipartimento di Progettazione Aeronautica
Via Claudio 21, 80125 Napoli - Italy
Web: www.dpa.unina.it/adag/
E-mail: coiro@unina.it ~ agodemar@unina.it

Paolo Leoncini
Italian Aerospace Research Centre (CIRA)
Via Maiorise - 81043 Capua (CE) - Italy
Web: www.cira.it
E-mail: p.leoncini@cira.it

KEYWORDS

Flight Accident Analysis, Flight Path Reconstruction, Flight Simulation, 3D Animations.

ABSTRACT

This paper presents a process of reconstruction, simulation, and advanced innovative visual animation of time dependent flight paths and aircraft evolutions. The approach is intended to be used in flight accident analysis. The possibility to produce advanced and realistic animations is today a valuable aid to investigators in analyzing an accident trajectory. Basic stages of the aircraft flight path analysis and reconstruction process will also be presented, discussing the benefits inherent to the proposed analysis method. The subject of this paper can be viewed as a typical application of what is known as "forensic engineering", i.e. the application of scientific and engineering knowledge to legal matters, such as accident reconstruction. Authors have created a well integrated simulation environment in order to make almost fully automatic the process of accident analysis.

INTRODUCTION

An extensive research has been carried out in the past 30 years on flight data treatment in accident analysis, and time dependent flight path/aircraft attitude reconstruction. Main contributions came out in the period between late 70's and middle 80's from researchers at NASA Ames Research Center. Bach in (Bach 1980) introduced a variational smoothing algorithm to be used in flight test and accident analysis and Bach and Wingrove in (Bach and Wingrove 1980), and in (Wingrove and Bach 1980) proposed a set of equations and procedures for determining aircraft motions from accident data.

Recently the increasing computer speed and elaboration power, coupled with the sophistication of modern computer graphics and 3D editing/rendering software, extended the benefits of these analysis methods by providing the possibility to produce advanced and realistic animations. These are today a valuable aid to investigators in analyzing accidents trying to understand possible causes. In fact, when a flight path is properly reconstructed and animated, it displays the sequence of key events and factors involved in an accident. Examples of advanced aircraft accident simulations and animations has been presented by Calkins in (Calkins 1994) under the form of case studies.

The subject of this paper can be viewed as a typical application of what is known as "forensic engineering", i.e. the application of scientific and engineering knowledge to legal matters, such as accident reconstruction.

Authors present here a process of reconstruction, simulation, and animation of flight paths based on a combination of methods proposed in the reported literature and coupled with advanced innovative visual animation techniques. In the following pages the basic stages in the process of aircraft flight path reconstruction are discussed, i.e. (*a*) the motion specification and estimation/simulation, (*b*) the geometric modeling and image rendering, (*c*) the animation and visual simulation, and (*d*) video production.

AVAILABLE DATA AND FLIGHT PATH RECONSTRUCTION

Data sources considered here, and commonly used in accident analysis practice, are (*i*) the ground-based Air Traffic Control (ATC) radar tracking systems or (*ii*) the on-board flight data recorders (FDR) along with VDR (Voice Data Recorder). A radar tracking system provides time histories of aircraft position (x,y), including altitude (z) (when the vehicle is equipped with an altitude transponder). Radar data are recorded by an Air Traffic Control (ATC) station in the vicinity of the accident site. An on-board flight data recorder (FDR) provides at least time histories of indicated airspeed (V), magnetic heading (ψ), barometric altitude (z), and normal body acceleration. FDR are carried by a wide variety of aircraft and are designed to withstand the rigors of severe accidents.

The reconstruction, and simulation of time dependent flight paths can be based on a combination of both ATC and FDR row data.

Fig. 1 presents the typical flow chart for processing accident data to obtain wind-axis forces and body-axis Euler angles. In the flow chart the classical dot-notation is used for time derivatives, and subscripts "b" and "W" stand for "relative to the body" (i.e. the aircraft) and "to the wind" respectively. The aircraft sideslip angle is denoted with β. Authors have implemented Fortran computer programs based on the method proposed in (Bach and Wingrove 1980) for the treatment of ATC and

FDR records. Radar position, heading, and altitude data recorded by ATC, in combination with pressure altitude, speed, heading, and load factor data recorded by on-board FDR, are first smoothed, reduced to same time interval and then used to derive estimated time histories that include airspeed, aircraft pitch (θ), roll (ϕ), and heading angles. Estimated quantities that are also recorded by FDR, such as the speed and the heading, are employed in double checking the quality and accuracy of the estimation process and they could also be used in attempting to estimate wind effects..

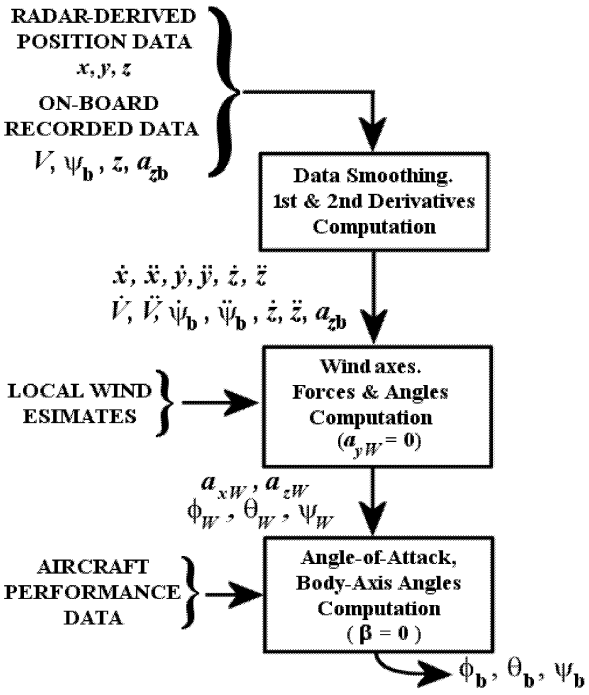


Figure 1: Accident Data Processing Flow Chart

Data treatment includes the correction of calibration errors and non-standard atmospheric conditions, providing time-history estimates of such parameters as ground speed, ground track, and flight path angles. Either data sources, in combination with meteorological and aircraft data, are utilized to estimate (*i*) aircraft wind-axis specific forces, i.e. excess thrust, and lift, and (*ii*) body-axis Euler angles, i.e. the orientation in space of the aircraft: roll, pitch, and yaw. Thrust and lift forces are important aircraft performance parameters, that, along with the Euler angle time histories, are the quantities needed in visualizing aircraft evolution along the accident trajectory.

SMOOTHING OF ATC & FDR DATA

Examples of ATC and FDR raw data are reported in fig. 2. Differences are visible between vehicle speed and heading derived from ATC track data (*x*- and *y*- time-histories) as opposed with those recorded by on-board FDR, which contain information related to aircraft relative wind. In fig. 3 and 4 are also reported examples of smoothed data.

The smoothing referred to in this paper consists of a procedure that smooths a flight data record and provides estimates of its first and second derivatives with respect to the independent variable, i.e. the time *t*. The algorythm used, adapted from (Bach and Wingrove 1980), passes a "least-squares moving arc" through the data. The data record consists of the samples z_j (j=1, … , *NPTS*) and the smoothing arc is a second-degree polynomial $x_i(t)$ spanning an interval of *NS* (odd) equally spaced data points such that

$$\begin{cases} x_i = a_0 + a_1(i-t) + a_2(i-t)^2 \\ t = j - (NS-1)/2 \end{cases}$$

where (a_0, a_1, a_2) are chosen to minimize the function

$$J = \sum_{i=j-NS+1}^{i=j} (z_i - x_i)^2$$

Values of the polinomial and its first two derivatives are computed at the central point p of each interval as the arc is passed through the data. Estimates near the beginning and end of the data are obtained by extrapolation of the first an last polinomials.

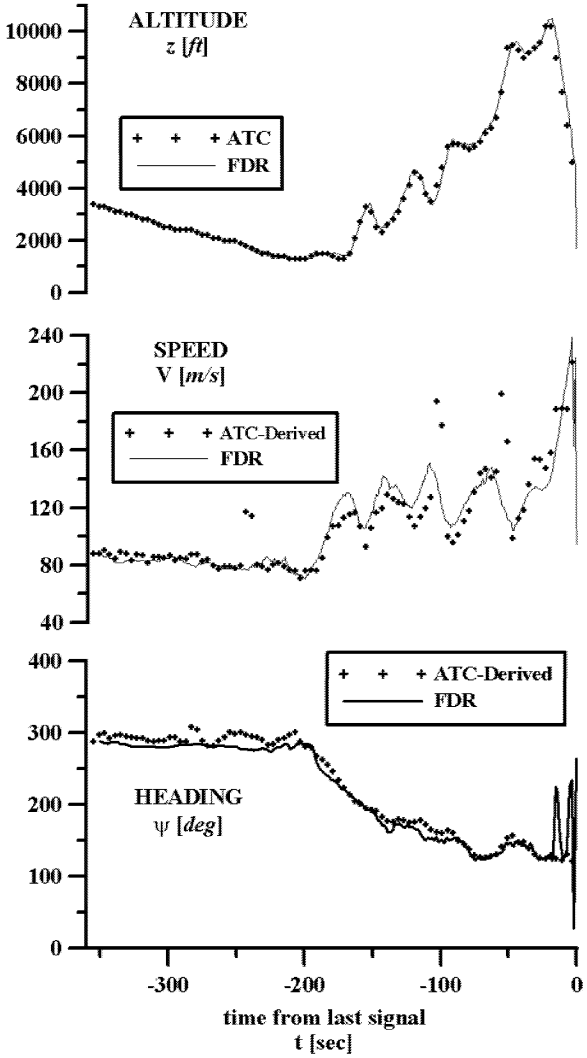


Figure 2: Example of ATC/FDR Raw Data

MOTIONS DERIVED FROM ACCIDENT DATA

One of the key steps in processing accident data indicated in fig. 1 is the estimation of the angle of attack by the evaluation of forces, airspeed and specific information about the airplane (lift characteristics). The angle of attack

allows the determination of the wind-axes, and of the body-axis Euler angle time-histories.

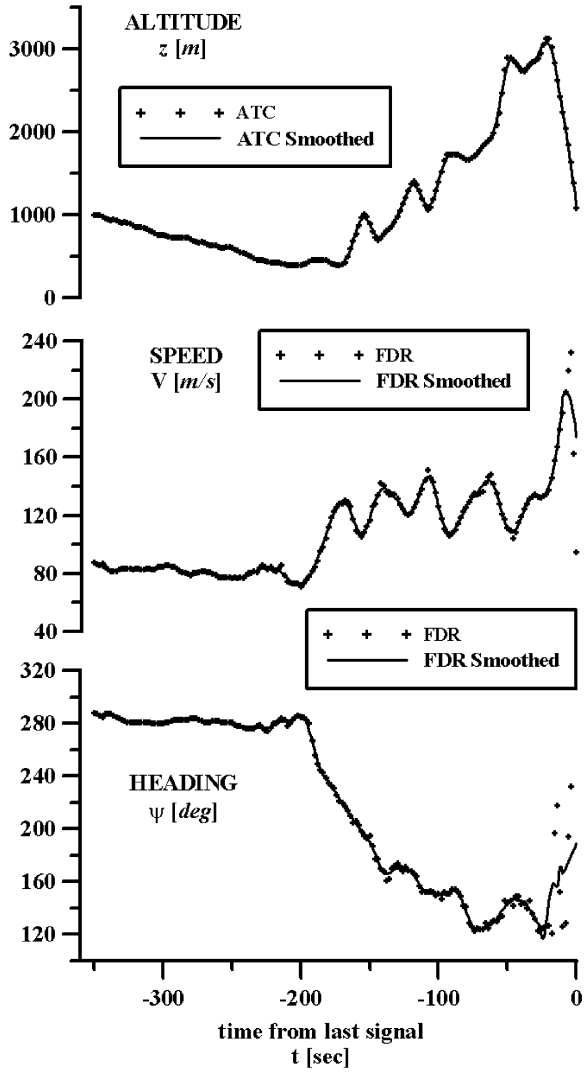


Figure 3: Example of FDR Raw Data (Dots) and Smoothed Signals (Continuous Line)

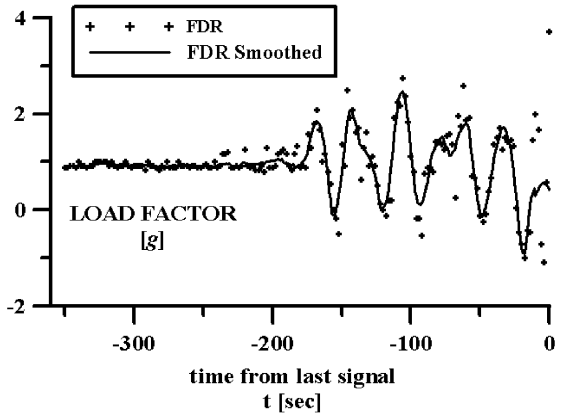


Figure 4: Example of Recorded and Smoothed Load Factor

The "vehicle-carried vertical frame" of reference F_V and the "wind-axis frame" F_W are introduced here, having their origins at the vehicle mass center and moving with it. Frame F_W has axis Ox_W along the velocity vector and axis Oz_W in aircraft plane of simmetry. The axes of F_V are parallel to a conventional stationary earth-surface frame F_E. Earth's curvature and rotation relative to an inertial frame are assumed negligible here. Definitions of the Euler angles and a review of important axis transformations can be found in (Bach and Wingrove 1980) and in (Etkin 1972).

Wind-axis Euler angles (ψ_W, θ_W, ϕ_W) with respect to frame F_E and the so-called "specific forces" (a_{xW}, a_{yW}, a_{zW}), i.e. acceleration vector components, may be determined from vehicle position derivatives and from the eventual wind information. Accordingly, the attitude of the "body frame" F_B relative to F_V is determined by Euler angles (ψ_b, θ_b, ϕ_b), which are of primary interest in the analysis of vehicle motion. Frames F_V and F_B are displaced by angle of attack α and sideslip angle β. In this development it is assumed a negligible sideslip along the flight path, which is consistent with the assumption of zero side specific force.

In motion reconstruction we assume that certain types of data anomalies, such as dropouts, wild points, etc., have been removed and that corrections for temperature, pressure, etc., have been made to air data in so far as possible.

Uncertainties are also admitted for meteorological data. In fact it can happen that wind measurements, if avaliable, may have been recorded many miles from the crash site and at a different time from that of the accident. Usually, wind data, i.e. wind velocity components (w_x, w_y, w_z), are obtained only as a function of altitude, and it must be often assumed a negligible vertical component, w_z=0. Despite these limitations, one can proceed to compute wind-axis Euler angles and forces along the trajectory, to estimate an angle-of-attack-time history, and to determine body-axis Euler angles.

The assumption of negligible side force (a_{yW}=0) and sideslip angle (β=0) is made here to obtain the desired solutions. There may be, of course, accident situations in which this assumption is not valid.

Assuming that time histories of vehicle position (x, y, z) and local winds (w_x, w_y, w_z) are available, they are first smoothed according to the procedure recalled in previous section to obtain estimates of vehicle velocity ($\dot{x}, \dot{y}, \dot{z}$) and acceleration ($\ddot{x}, \ddot{y}, \ddot{z}$). Wind-axes Euler axes and airspeed are then determined according to the formulas

$$\psi_W = \tan^{-1}\left[\left(\dot{y} - w_y\right)/\left(\dot{x} - w_x\right)\right]$$
$$\theta_W = \sin^{-1}\left[-\left(\dot{z} - w_z\right)/V\right]$$
$$V = \left[\left(\dot{x} - w_x\right)^2 + \left(\dot{y} - w_y\right)^2 + \left(\dot{z} - w_z\right)^2\right]^{1/2}$$

The "excess thrust" a_{xW} (i.e. the difference between available thrust and aircraft drag force, per unit mass) is next determined by the formula

$$a_{zW} = C_2 \cos\phi_W - C_1 \sin\phi_W$$
$$\phi_W = \tan^{-1}\left(C_1 / -C_2\right)$$

where

$$C_1 = \left(\ddot{y}\cos\psi_W - \ddot{x}\sin\psi_W\right)$$
$$C_2 = \left(\ddot{z} - g + a_{xW}\sin\theta_W\right)/\cos\theta_W$$

and g is the gravity acceleration. Notice that in the above equations there is no dependence on wind-time-derivative information. Finally, assuming a linear dependence of lift force from the angle of attack

$$\alpha = \alpha_0 - m \cdot a_{zW} / Q \cdot S \cdot C_{L\alpha}$$
$$Q = \frac{1}{2}\rho V^2 \quad (1)$$

where m is the aircraft mass, S the wing area, ρ the air density, Q the flight dynamic pressure, and $C_{L\alpha}$ the derivative of the lift coefficient with respect to α. Values of α_0 and $C_{L\alpha}$ depend primarily on aircraft configuration and Mach number and are tabulated for a given aircraft. At high angles of attack, when the above linear dependence is no longer adequate, a flate-plate relationship yelds a good approximation

$$\alpha = \tan^{-1}\left(a_{xW} / a_{zW}\right) \quad (2)$$

Once α is estimated, the body angles are determined by the formulas

$$\begin{aligned}
\psi_b &= \psi_W + \tan^{-1}\left[\sin\alpha \cdot \sin\phi_W \cdot \right. \\
&\quad \left. \cdot\left(\cos\alpha\cos\theta_W - \sin\alpha\cos\phi_W \sin\theta_W\right)^{-1}\right] \\
\theta_b &= \sin^{-1}\left(\cos\alpha\sin\theta_W + \sin\alpha\cos\theta_W \cos\phi_W\right) \quad (3) \\
\phi_b &= \tan^{-1}\left[\cos\theta_W \sin\phi_W \cdot \right. \\
&\quad \left. \cdot\left(\cos\alpha\cos\theta_W \cos\phi_W - \sin\alpha\sin\theta_W\right)^{-1}\right]
\end{aligned}$$

While all above quantities can be derived in principle by ATC data, the following one are derived by FDR in order to complete the accident scenario. An FDR records, consisting of V, ψ_b, z, and $-a_{zb}$ time-histories, along with time derivatives $\dot{V}$, $\dot{\psi}_b$, $\dot{z}$, and $\ddot{z}$, wind information, and aircraft performance data, can be utilized to provide an accident scenario.

First, wind-axis pitch angle is determined by the formula

$$\theta_W = \sin^{-1}\left[-\left(\dot{z} - w_z\right)/V\right]$$

Next, the forces a_{xW} and a_{zW} and roll angle ϕ_W are estimated by using equations

$$\begin{aligned}
a_{xW} &= \dot{V} + \left(g - \dot{w}_z\right)\cdot\sin\theta_W + \\
&\quad + \left(\dot{w}_x\cos\psi_W + \dot{w}_y\sin\psi_W\right)\cdot\cos\theta_W \\
a_{zW} &= C_2\cos\phi_W - C_1\sin\phi_W \\
\phi_W &= \tan^{-1}\left(C_1 / -C_2\right)
\end{aligned}$$

where

$$\begin{aligned}
C_1 &= \dot{\psi}_w V\cos\theta_w + \dot{w}_y\cos\psi_w - \dot{w}_x\sin\psi_w \\
C_2 &= \left(\ddot{z} - g + a_{xw}\sin\theta_w\right)/\cos\theta_w
\end{aligned}$$

Notice now that in these formulas there is dependence from wind time-derivatives, estimates of which are generally unreliable or unavailable. Since w_z is usually not measured and othe two components are recorded only as a function of altitude, the best estimates of the wind derivatives are

$$\dot{w}_x = \frac{dw_x}{dz}\dot{z}, \qquad \dot{w}_y = \frac{dw_y}{dz}\dot{z}, \qquad \dot{w}_z = 0$$

Also the factor $\dot{\psi}_W$ must be approximated by $\dot{\psi}_b$, which is included in the data set, or can be derived by ATC and wind data. The angle of attack is estimated in the same way as above.

Finally, the body-axis Euler angles are determined by the same equations (3)

$$\begin{aligned}
\psi_W &= \psi_b - \tan^{-1}\left[\sin\alpha\cdot\sin\phi_W \cdot \right. \\
&\quad \left. \cdot\left(\cos\alpha\cos\theta_W - \sin\alpha\cos\phi_W\sin\theta_W\right)^{-1}\right] \\
\theta_b &= \sin^{-1}\left(\cos\alpha\sin\theta_W + \sin\alpha\cos\theta_W\cos\phi_W\right) \\
\phi_b &= \tan^{-1}\left[\cos\theta_W\sin\phi_W \cdot \right. \\
&\quad \left. \cdot\left(\cos\alpha\cos\theta_W\cos\phi_W - \sin\alpha\sin\theta_W\right)^{-1}\right]
\end{aligned}$$

The FDR record contains the apparently redundant–a_{zb} quantity. This redundancy is utilized to provide an indipendent estimate of angle of attack according to

$$\alpha = \alpha_0 - m \cdot a_{zb} / Q \cdot S \cdot C_{N\alpha}$$

similar to (1), where $C_{N\alpha}$ is the derivative of the normal force coefficient. Again, α_0 and $C_{N\alpha}$ would be found as tabulated functions of configuration and flight Mach number for a given aircraft. Outside the linear region the flate-plate relation equivalent to (2) is

$$\alpha = \tan^{-1}\left(a_{xW} / a_{zb}\right)$$

This estimates of a gives the wind-axis reaction term a_{zW} via the following formula

$$a_{zW} = a_{zb} / \cos\alpha - a_{xW}\tan\alpha \quad (4)$$

indipendently of ψ_W. Redundancy implied by (4) provides a check on the assumption that ratio (a_{yW}/a_{zW}) is negligibly small. In fact it is given by

$$\left|a_{yW} / a_{zW}\right| = \left[\left(C_1^2 + C_2^2\right)/ a_{zW}^2 - 1\right]^{1/2}$$

Redundancy can also be used to compensate anomalies in the data. On-board instruments are subject to unusual operating conditions during an accident, and portion of the recovered records may contain significant errors. If the altimeter is suspect, the roll angle ϕ_W can be computed from

$$\phi_W = \sin^{-1}\left(-C_1 / a_{zW}\right) \quad (5)$$

When data from the directional gyro are considered unreliable, the magnitude of ϕ_W can be determined from the expression

$$\left|\phi_W\right| = \tan^{-1}\left[\left(a_{zW} / C_2\right)^2 - 1\right]^{1/2} \quad (6)$$

The alternate solutions for roll angle given by equations (5) and (6) are used also to provide data-consistency check of the four FDR measurements.

As a final observation, it can be shown that the following expression is valid

$$C_2 = a_{zW}\cos\phi_W$$

from which an expression for the vertical acceleration can be worked out

$$\ddot{z} = a_{zW}\cos\phi_W\cos\theta_W - a_{xW}\sin\theta_W + g \quad (7)$$

Hence (7) may be integrated twice, and with appropriate choice of constants, the result should match the altitude record.

Fig. 5 and 6 show examples of estimated angle-of-attack, body-axis roll and pitch, and body-load-factor time-histories.

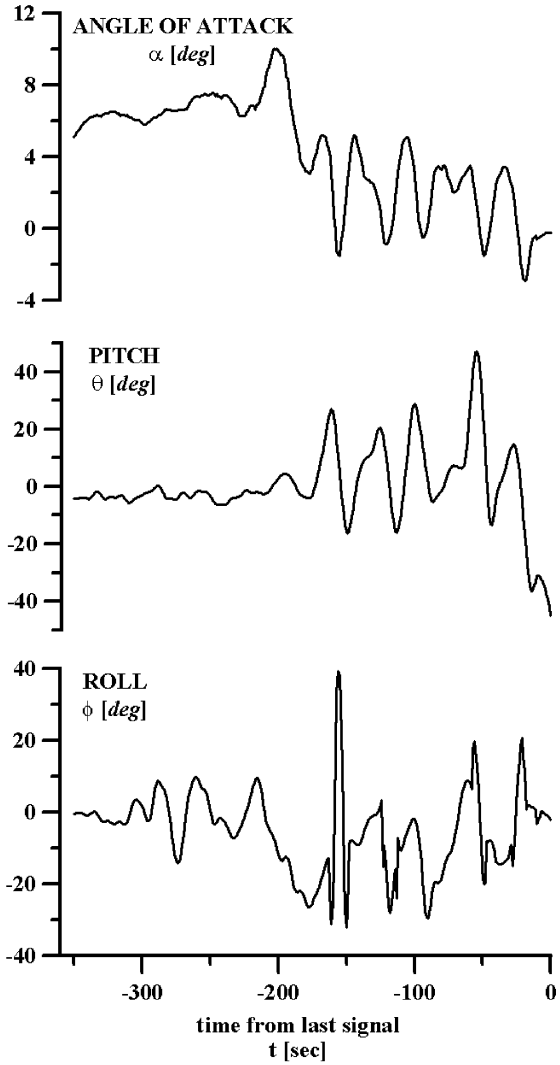


Figure 5: Example of Estimated Time Histories

FLIGHT VISUAL ANIMATIONS & SIMULATIONS

Once all the data have been corrected and smoothed and the necessary flight parameters have been estimated and crosschecked, one can proceed to detailed scenarios inspections and analyses. Optimised video sequences, and clearer animations have been created by authors using the well-known *3d studio max* (*3ds-max*) software and scripting capabilities. *3ds-max* is programmable via his native *Max-Script* interpreter. User is able to read from file relevant geometric entities, such as aircraft center-of-gravity positions along the flight path and aircraft space orientations (stored according to a given time frequency), and to manipulate a given CAD object, in this case the 3D aircraft model, along the time-line. Native key-framing techniques are available in *3ds-max* to edit and optimize the complete animation. Advanced rendering techniques give then to the animation a realistic outcome with their sophisticated shading, shadows, transparency and texture effects.

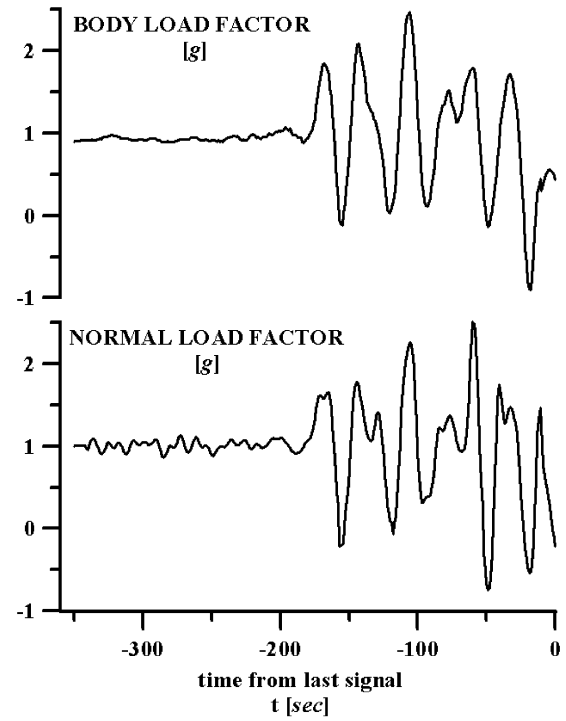


Figure 6: Estimated Load Factor Time Histories

Fig. 7 represents a possible flight accident scenario. Conventional earth axes are depicted along with the airport runway. Some critical flight phases before the unrecovered dive are indicated on the flight path.

In fig. 8, 9, 10 are reported superimposed frames of some interesting flight phases. Effectiveness of such a tool is evident, from flight path and manoeuvres inspection, fig. 6 and 7, to closer detail analysis, such as in fig. 8 representing a typical case of "pilot induced oscillation" (PIO).

Visual flight simulations are achieved with the aid of the open-source flight simulator *FlightGear* (*FG*), presented in (Sehgal, Deters and Selig. 2002). In this context *FlightGear,* which is designed also to accept flight data from external numerical models, is used as a visualization server, while a client program (written in c++ and compiled against some *FG* base-libraries) passes in real-time via an UDP net protocol all the smoothed and estimated quantities that are needed to have *FG* visualize the flight. The user is able to choose the visualization mode that best fits his needs, such as view from inside the cockpit, with or without instruments or head-up-display (HUD), view from an outside camera following the airplane (with or without HUD, see fig. 11), or view from a fixed outside camera pointing to the aircraft moving in the scene.

Reconstructions are then enhanced by synchronizing visual animations with dialogues between pilots and ATC, available via on-board voice data recorders. This approach is particularly helpful when data sets provided by FDR are made of a small number of flight parameters. As an example, on smaller commercial aircraft thrust data could be unavailable. Such a lack of information, coupled with a poor collection of data on local wind conditions, could be significant in the analysis of complex manoeuvres when the physical sense time-histories is not straightforward.

An ultimate aid, especially from the forensic engineering perspective, is the synchronization of animation and dialogues with subtitles. This can be achieved efficiently by launching a flight simulation contextually with one of the available freeware subtitler software or by post-producing the video animation with professional video editing equipment.

Examples of flight simulation synchronized with subtitles are presented in fig. 11: the upper part of each screenshot is a *FlightGear* window while the lower part is the subtitler window. If VDR (Voice Data Recorder) data are available, these can be listened at same time with the phrases appearing in the subtitle windows. Any comments or note to the voice data (some times it is necessary to draw the attention to some specific instant during the accident) can appear in this window.

Authors have created a well-integrated simulation environment by tuning each other all the tools that they use in order to make almost fully automatic the process of accident analysis.

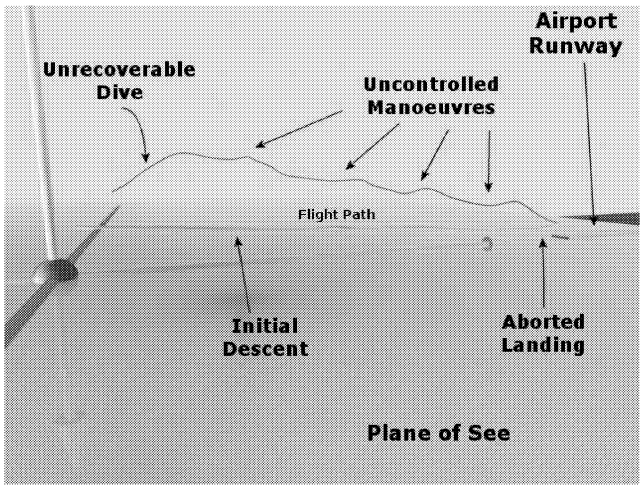


Figure 7: A Possible Accident Scenery

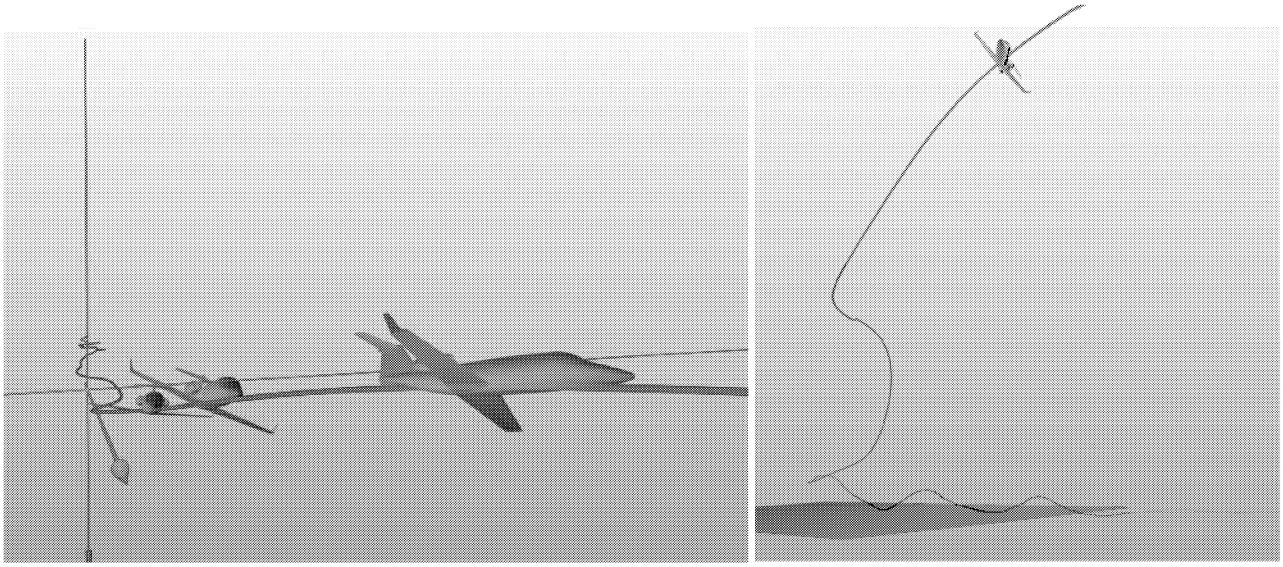


Figure 8: Wave-Off Manoeuvre and Flight Path 3D Visualization

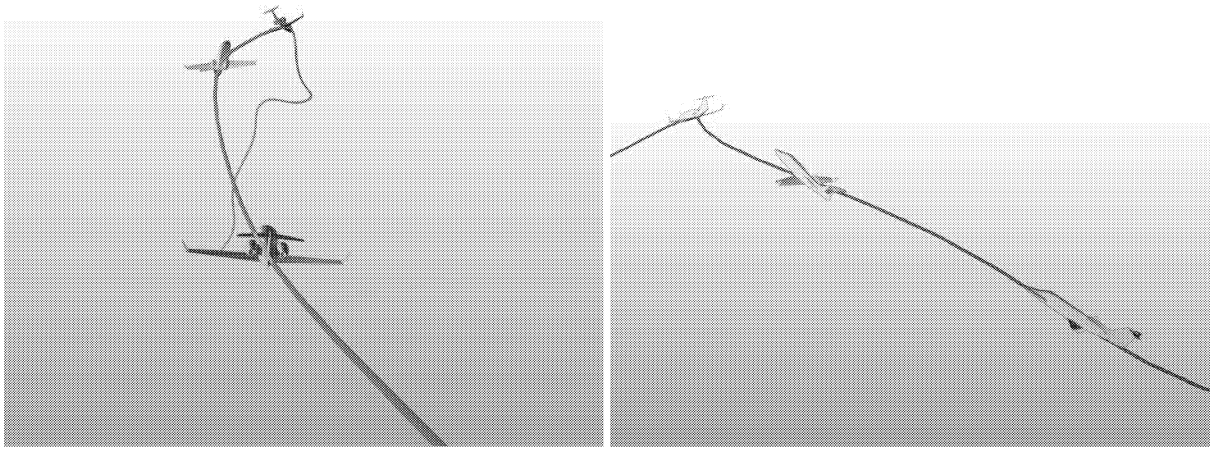


Figure 9: Examples of Flight Path and Manoeuvre Inspections

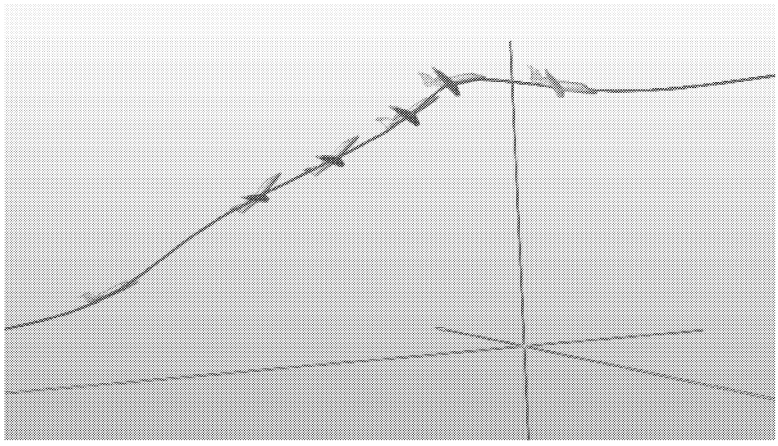


Figure 10: Examples of Pilot Induced Oscillations (PIO)

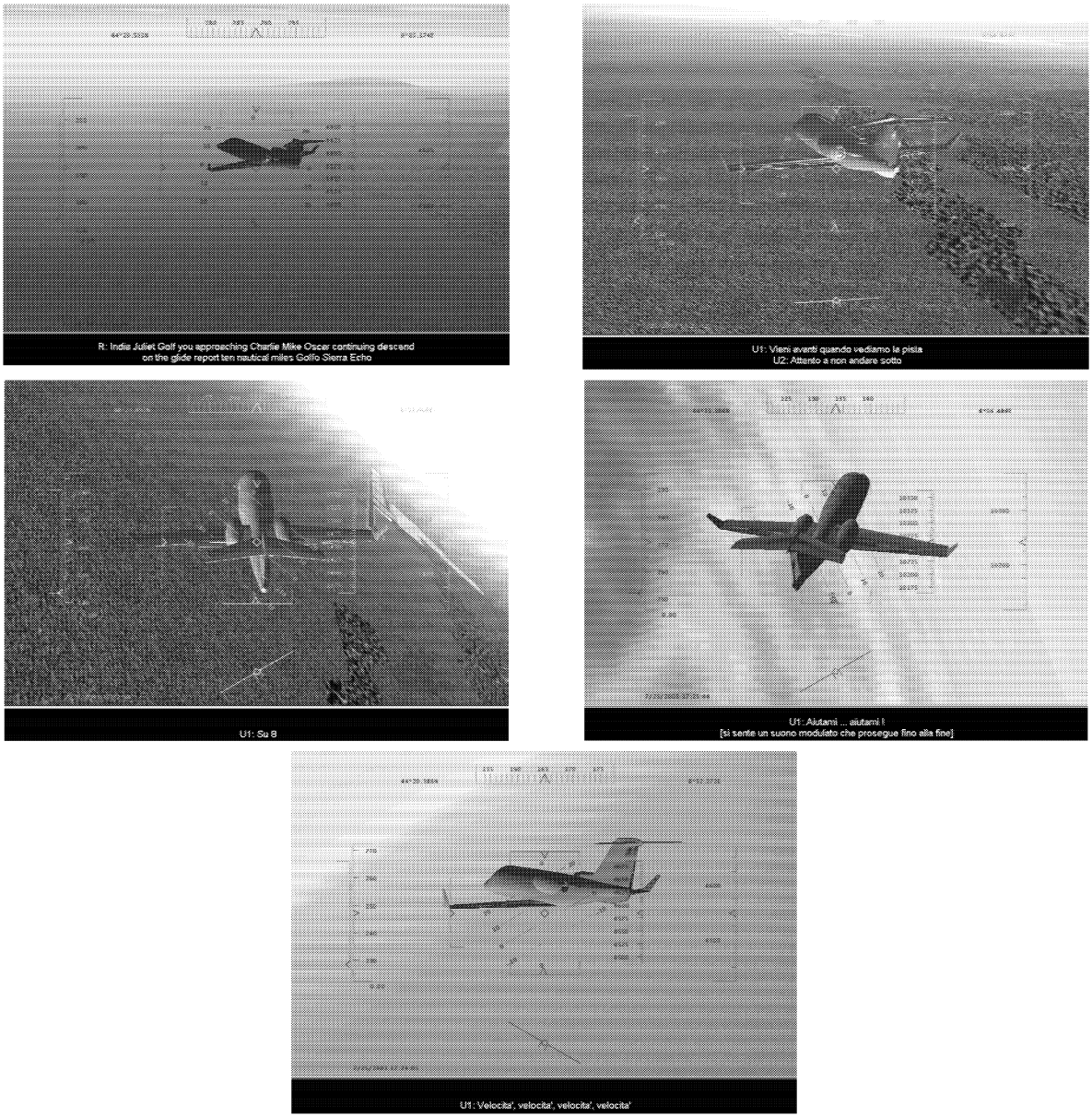


Figure 11: Flight Simulation Screenshots with Subtitles

CONCLUSIONS

Flight accident reconstruction and visualization has been performed using modern techniques. The whole procedure made of FDR (Flight Data Recorder) data, their analysis, flight motion reconstruction, voice reproduction and different possible way of visualization have been shown. The necessity of having available different tools to represent in different ways the motion of the aircraft to reconstruct the accident arises from the type of visualization that normally is addressed to different kind of interlocutors such as technicians, pilots, lawyers and judges. The high level of generality and modularity of the developed software, allows the representation and the analysis of any accident data in short time.

REFERENCES

Bach, R. E. Jr. 1980. "A Variational Technique for Smoothing Flight Test and Accident Data". *Journal of Aircraft*, Vol. 19, No. 7..

Bach, R. E. Jr. and Wingrove, R. C. 1980. "Equations for Determining Aircraft Motion from Accident Data". NASA TM-78609.

Calkins, D. E. 1994. "Aircraft Accident Flight Path Simulation and Animation". *Journal of Aircraft*, Vol. 31, No. 2.

Etkin, B. 1972. Dynamics of Atmospheric Flight. John Wiley & Sons. Chapters 4 and 5.

Sehgal, B., Deters R. W. and Selig M. S. 2002. "Icing Encounter Flight Simulator". AIAA Paper 2002–0817. 40th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

Wingrove, R. C. and Bach, R. E. Jr. 1980. "Analysis of General Aviation Accidents Using ATC Data Records". *Journal of Aircraft*, Vol. 20, No. 10.

AUTHOR BIOGRAPHY

Domenico P. Coiro is currently associate professor at Dipartimento di Progettazione Aeronautica, (Dept. of Aeronautical Engineering) of the University of Naples "Federico II". His research and teaching activities have been devoted over the years in the fields of Aerodynamics, Applied and Computational Aerodynamics, Vehicle Flight Dynamics.
In 1986/87 he had been Teaching Assistant at Department of Aerospace Enginnering, Pennsylvania State University, State College, USA. From 1987 to 1990 he had been in charge as Head of Subsonic/Transonic department of CIRA, Italian Center for Aerospace Research. He currently teaches Flight Dynamics course at the Faculty of Engineering, University of Naples, Italy, where he has been chairman of numerous degree theses.
He has presented numerous scientific memories at national and international congresses some of which have been published on national and international technical journals. He has also matured a certain experience in the field of the courses of professional formation, having developed some for account of the ALENIA, the Italian main aerospace industry.
Within the Aircraft Design & Aeroflightdynamics Group (ADAG) he contributed to the developement of the *JDYNASIM* project. It is a flight simulator available on the web, developed at DPA, and recently rewritten in Java language, in order to allow internet users to perform remote interactive flight simulation sessions of the aircraft flight.
In the field of flight mechanics Professor Coiro works on the development of numerical codes for the prediction of aircraft performances for particular configurations and particular conditions. He has carried out researches on light aircraft design and sailplane design. Since 1993 he is involved in research activities about the design of ultra-light aircrafts (ULM)
Professor Coiro also works in the field of renewable energy where he has carried out some researches on vertical and horizontal axes wind-turbines analysis and optimization.

Agostino De Marco, is an Aerospace engineer with a PhD in Naval engineering from the University of Naples "Federico II". Since 2002 he has a position of Researcher at the Department of Aerospace Engineering of the University of Naples "Federico II" where he is involved in the Aircraft Design & Aeroflightdynamics Group (ADAG) research activities, his main research areas are: Flight Mechanics, Flight Simulation, Aircraft Design, and Computational Aerodynamics. Related fields of his interest are also: Fortran and C/C++ Programming, Scientific Computing, Grid Generation and CFD, Flow Visualization, CAD and Solid Modeling.

Web pages

Department of Aerospace Engineering, University of Naples (DPA):	www.unina.it	(English/Italian)
Aircraft Design & Aeroflightdynamics Group (ADAG):	www.unina.it/adag/	(English)
Domenico P. Coiro:	www.dpa.unina.it/coiro/	(English/Italian)
Agostino De Marco:	www.dpa.unina.it/demarco/	(English)

SIMULATED STROBOSCOPIC ILLUMINATION FOR UNSYNCHRONIZED MOTION DYNAMICS

Zhaoyi Li
Renate Sitte
Faculty of Engineering and Information Technology
Griffith University, Gold Coast, Australia
Zhaoyi.Li@griffith.edu.au
r.sitte@griffith.edu.au

Keywords: Physics based modeling, scientific visualization, Micro-Electro-Mechanical systems, visual simulation

Abstract: This paper discusses a simulation method that can mimic objects that move in a virtual world in very fast and slow motion, so that their dynamics can be observed in truthfully time scaled visualizations. This poses a series of problems for truthfully time scaled visualizations, including hardware constraints. A simple truthful time scaling is not sufficient because it may result in one of the objects still at non-observable fast motion, while another practically reaches a standstill. Our objective is to incorporate this into a CAD simulation environment that aids in the design of Micro-Electro-Mechanical Systems (MEMS). Those movable parts in MEMS can flick, pump or pulsate at rates of up to 500 Hz, while other components in the same MEMS move at much slower speeds. To maintain the relative speeds, we simulate a stroboscopic illumination on the scene of moving objects. This is effectively filtering the images, and displaying only a subset at a rate such that they become visible and observable, while maintaining physical truthfulness. In this paper we present the results of a systematic combinatorial analysis, including hardware constraints, for finding the range in which stroboscopic simulation works well, and for which proportions it is not necessary.

1. INTRODUCTION

In areas of visual simulation, the truthful visualizations of dynamics are rather limited to scientific visualizations, while there are some visualizations of high-speed air or liquid (Kihm et al. 1998), truthful visualizations of unsynchronized dynamics for mechanic objects are not in abundance (Sitte 2003). However there is an increasing demand for truthful visualizations of dynamics, for the truthful visualizations may help to reduce a design trial and error phase for system elements dimension and placement. Our objective is to incorporate VR visualizations of functioning devices into our ongoing CAD simulation environment project for Micro-Electro-Mechanical Systems (MEMS) (Li 2001). The research focuses on detecting design flaws or timing issues of the components of MEMS being designed.

MEMS are miniaturized mechanical or optical equipment less than a millimetre in size. While some have movable parts, such as membranes, valves, cantilevers, others may have only fixed parts. Problems arise in time scaled visualizations. In a mechanic environment it is possible for one component to move at 500Hz while the others move at a rate of 100 times slower. To display simultaneously such components in action in a CAD-VR environment is a challenge.

The trivial solution is downscaling in time to slow motion. This does not work when we have asynchronous events being displayed, e.g. one very fast, and one very slow, because the slow one would come to a standstill or be distorted. To the observer the relative movements between the two bodies may no longer be truthful. Motion perception is a complex issue, involving brain activities, sometimes illusion occurs, like motion aftereffects (Goldstein, 1996), another example is that, the high-speed gear appears rotating counter-clockwise when actually it rotates in clockwise direction. This gives a challenge for visual simulation of fast movement.

Motion perception research has a long history. Stroboscopic motion, the motion perceived when two lights are presented at different times and different locations (Zeki 1992) has been largely used in movies and televisions. If two lights are flashed on and off in succession, with an appropriate interval between them, the viewer perceives one light that appears to move from the position of the first light to the position of the second, this is then apparent motion.

There are many factors that determine the effect of stroboscopic motion, including shape, size, color, texture of the object. Three factors are particularly relevant to computer animation: (1) the time interval between projection of the separate displays; (2) the light intensity of the displays; and (3) the spatial distance between each of the displays (Schiffman 1995). The "Korte's law" states that apparent motion can only result when these three factors are properly synchronized (Gepshtein 2001).

Stroboscopic motion has a critical threshold of about 16 frames per second in order for it to be perceived as smooth

and continuous. But 24 frames per second can make a "flicker" free animation.

Wertheimer found that if the intensity of the lights and distance between them are kept constant, then what is perceived depends on the interval of time between the flashing of the two stimulus ("ISI": inter-stimulus interval). (a)When the ISI is 200 msec or longer, subjects report seeing two lights flash in succession, at different places. (b)When the ISI is about 60 msec, subjects "see" one moving object; it appears to move from the location of the first light, to the location of the second light; (c)When the ISI is shorter than 60 msec, subjects may perceive "phi" or "objectless" movement – a "pure" sensation of movement, without a concomitant perception of moving objects. (d)When the ISI is zero (so that the lights flash simultaneously), they are perceived veridically- i.e., observers perceive the two flashing light as one moving light (Wertheimer 1999).

While these researches study the stroboscopic perception for not very fast motion. This paper investigates stroboscopic effect for displaying very fast and slow movement simultaneously in an easier observable image, while maintaining physical truthfulness. Besides time interval, distance moved between displays is another important factor for visual effect considered in this paper, as fast and slow motions are both simulated. The method for modeling unsynchronized motion dynamics for truthful visualizations is discussed in section 2. Section 3 has an analysis of the systematic combinatorial simulation results, including the system requirements of the simulation, series of visualization images, and the analysis of the simulation results. Section 4 concludes this paper and discusses the future work.

VRML is used as programming language for the graphical part, because the virtual reality toolbox based on VRML is incorporated with MATLAB, The MEMS virtual prototyping software that we are developing, uses Matlab as its design environment that is typically used by Engineers.

2. TRUTHFUL UNSYNCHRONIZED DYNAMICS MODELING

To overcome the difficulties of presenting simultaneously fast and slow movements on the screen, we simulate a stroboscopic illumination (Li 2003) on the moving objects. This is effectively filtering the images, and displaying only a subset at a rate such that they become visible and observable, without sacrificing their relative movement.

There is a great variety of moving structures in MEMS such as flicking cantilevers, pumping membranes, etc. all at possible very high frequency, while others are several orders of magnitude slower. We have chosen the challenge of two gears rotating at different, specified speed, but displaying their displacement only at the stroboscopic illumination intervals (SII), considering that rotation is likely to pose additional visual challenges.

In stroboscopic simulation, there are two main time components, one is the stroboscopic illumination interval (SII), during which the object moves at its own speed, and another is the stroboscopic flash duration (SFD), during which the stroboscopic light is applied, and the object appears to stop as shown at the position at the SII. As an analogy, one can think of a square signal, where the high is the SFD and the low is the SII.

The total time for a cycle of a stroboscopic flash consists of two parts: the time of SII (t_{SII}), and the time of SFD (t_{SFD}):

$$T_{cycle} = t_{SII} + t_{SFD} \quad (1)$$

The position P_n of the object after n stroboscopic cycles is recalculated as

$$P_n = n * t_{SII} * f \quad (2)$$

where f is the frequency of rotation of the object.

Another time parameter is the elapsed time, which is the total time for which the object moves, not including the time for SFD. This is calculated by

$$T_{ElapsedTime} = n * t_{SII} \quad (3)$$

The unsynchronized motion dynamics are modeled in truthful VR visualizations. The objects are shown at the intervals (SII) during the flash period (SFD). The SFD is used for the visual purpose, making the object in high speed visible and observable. The SII and SFD can be set at different rates, and determining a suitable SII and SFD is part of this study.

3. ANALYSIS OF COMBINATORIAL SIMULATION RESULTS

This section first describes the importance of the computer capability to simulation, followed by the systematic combinatorial experiment design and the visualization images, finally an analysis of the simulation results was made.

3.1 System Performance

Computer capability is a very important factor to affect the simulation results. The progress in computer hardware and software has made a pervasive entrance on modeling and simulation. The incredible advance in computer hardware is acknowledged as making simulation a viable problem-solving technique for some and the preferred technique for many (Nance 2002). The simulation can have better results on a computer of good performance. System capability includes the CPU speed, memory, monitor frame rate, color setting and pixel setting, etc. All these can affect the experiment results. The simulation effects are changed even when the color

setting is changed to 'true color (24 bit) from the previous setting of 'true color (16 bit)'. Our simulation is done on a normal setting computer, which would correspond to a realistic environment of typical users of our MEMS-CAD package. The settings are as follows:

Table 1: System Setting

CPU speed	Pentium III 733 MHz
Memory	128 MB
Graphic card	Intel810
Frame rate	75 Hertz
Pixel setting	800x600
Color setting	16-bit true color

For short time of interval, and high-speed objects, there is a high requirement of the computer system, a high-speed CPU for running the program, and a high-speed graphic card for transmitting the data to the graphic process. Otherwise the real time animation cannot be realized.

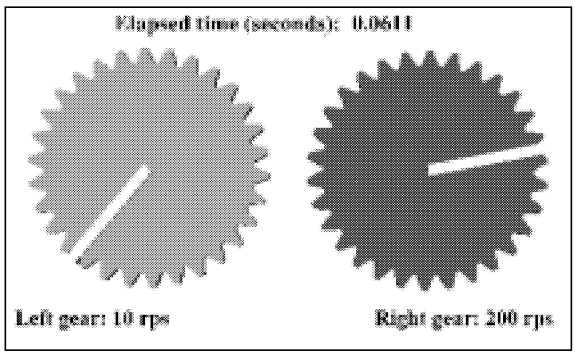
The CPU usage is checked before and after our simulation program is run. Some tested data for CPU usage is as follows in Table 2:

Table 2: CPU Usage

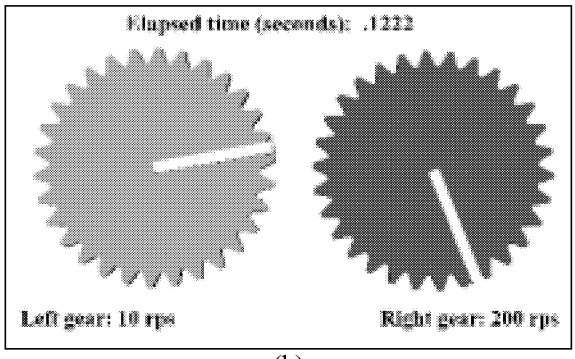
Stroboscopic flashes	CPU usage before running	CPU usage after running
55 flashes	4%	7%
12 flashes	2%	2%
66 flashes	3%	0%
41 flashes	8%	7%
53 flashes	4%	2%
23 flashes	4%	3%
83 flashes	1%	13%
673 flashes	5%	2%
15 flashes	2%	5%
5 flashes	5%	43%
3 flashes	6%	23%
3 flashes	6%	39%
3 flashes	3%	72%
5 flashes	2%	5%
10 flashes	4%	10%

3.2 Combinatorial Experiment

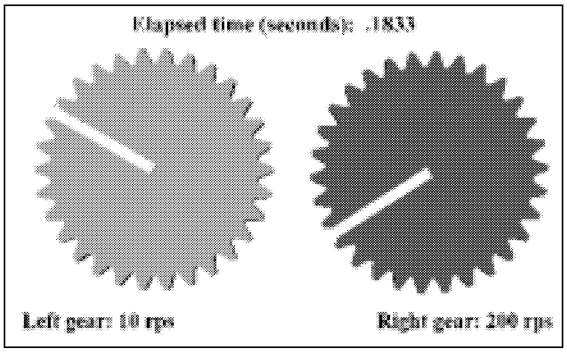
To study the effect and suitability range of the simulated stroboscopic illumination, a combinatorial study was performed, by varying systematically the speeds of two gears in specific and different increments. For each combination the simulated Stroboscopic Illumination Interval (SII) and the Stroboscopic Flash Duration (SFD) was simulated. We did this following the *physics based simulation* paradigms, maintaining timing proportions truthfully.



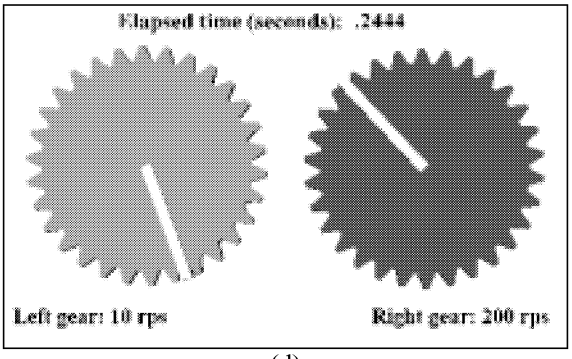
(a)



(b)



(c)



(d)

Figure 1: Visualization Images of Stroboscopic Movement of Different Speed Gears in Series (a) at start position; (b) 1 step; (c) 2 steps; (d) 3 steps

In our visual simulations we used two gears with outer radius 1.6 unit (in VRML), inner radius 1.4 unit, with 30 teeth. A white mark was placed on the gears for easier observation. In our combinatorial experiment we increased the rotation speed of one gear in smaller steps from 10 rps (revolutions per second) to 100 rps; and the second gear in larger steps from 100 rps to 500 rps. To find out suitable and pleasant SFD with properly synchronized SII we tested by changing the setting of the SFD and SII. The results are explained in the next section.

An example applying the stroboscopic simulation to moving gears is shown here. Figure 1 shows a snapshot of the two gears rotating in clockwise direction with the left at 10 rps, and the right one at 200 rps. The marks on the gears help to see the moving gears. Both gears started with their mark reset to a 12 o'clock position. These series of stroboscopic pictures shows the gears at the position of 0, 1, 2, 3 steps of SII at 0.0611 seconds. The reason for setting SII at 0.0611 seconds is to avoid displaying it in a multiple of the step of the next SII position, which would overlap with a previous position, and make it appear as staying always on the same place, instead of advancing in the rotation.

3.3 Analysis of Simulation Results

Table 3 is a sample table of the simulation results of SFD at 0.03 seconds with synchronized SII at 0.02 seconds for the combination of gears rotating at different speeds. The first column shows the speed for the first gear, and the first row shows the speed for the second gear. The number shows different levels of effects as follows:

Score 5: (very good), the gears run in stroboscopic MODE smoothly and in right direction (in clockwise direction), the rotation appears realistic and running in high speed, and the difference in number of illumination flashes and intensity of the marks makes the difference in speed of two gears looks obvious; Score 4: (good): the stroboscopic movement is in the right direction, but the rotation is not very smooth, the animation images still shows the difference in speed of two gears in an obvious truthful effect; Score 3: the rotation runs smoothly and in correct direction, but the number of illumination flashes and the intensity of the rotating marks do not make much difference between two gears, giving the illusion that the two gears are not much different in speed. Score 2: the rotation becomes more jumpy and the fast and slow movement can not be easily observed. Score 1: (backward move), the rotation of one or two gears looks as in the wrong direction (counter clockwise direction) and becomes more jumpy.

The number in between in the table can be the results in between, e.g., 4.3 shows the results between 4 and 5, 4.5 also shows the results between 4 and 5, but has a better results than 4.3.

Table 3: Simulation Effects for SFD at 0.03 seconds and SII at 0.02 seconds

	100	150	200	250	300	350	400	450	500
10	4	4	4	4	4	4	4	4	4
20	4	4	4	4	4	4	4	4	4
30	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5
40	4	4	4	4	4	4	4	4	4
50	4	4	4	4	4	4	4	4	4
60	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5
70	4	4	4	4	4	4	4	4	4
80	4	4	4	4	4	4	4	4	4
90	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5	2.5
100	5	5	4.8	4.8	5	4.8	5	4.8	5

Figure 2 on next page is a series of graphs showing the simulation effects graphically for the systematic combinatorial experiment results at different SFD with synchronized SII, with lighter color for better effects. From the table and graphs, we know that the range of SFD between 0.03 seconds to 0.1 seconds with synchronized SII provides good stroboscopic movement images. Good stroboscopic animation effects can only result when SFD and SII are properly synchronized. It is found that, to synchronize SFD with SII properly the SII should be as same time as or a little bit less than the period of SFD. The set of data for the five graphs is respectively (a) SFD=0.03 seconds, SII=0.02 seconds; (b) SFD=0.04 seconds, SII=0.02 seconds; (c) SFD=0.06 seconds, SII=0.04 seconds; (d) SFD=0.08 seconds, SII=0.06 seconds; (e) SFD=0.1 seconds, SII=0.06 seconds. The SFD at 0.03 seconds with SII at 0.02 seconds provides best animation results for the majority of combinations of different speed of the gears. The stroboscopic rotation images show that there are more illumination flashes for high speed gear than for low speed gear, and the intensity of the rotating mark on the higher speed gear is lighter than that of the mark on the low speed gear, giving the illusion that high speed gear rotates fast.

Comparatively shorter SFD and SII provide smooth and continuous stroboscopic images, however longer SFD and SII provide better images for gears rotating at large range of speed, showing the fast and slow movement simultaneously at better effects. For low speed gears, longer SFD and SII provide good animation, because shorter SFD creates displays at successive SFD simultaneously. For high speed gears, shorter SFD and SII provide good animation, as shorter SFD will not provide displays for successive SFD simultaneously because of the high speed. For example SFD of 0.04 seconds produces slightly better results for 100-500 rps than for 10-100 rps. SFD of 0.1 seconds has better results for 10-100 rps than for the 100-500 rps.

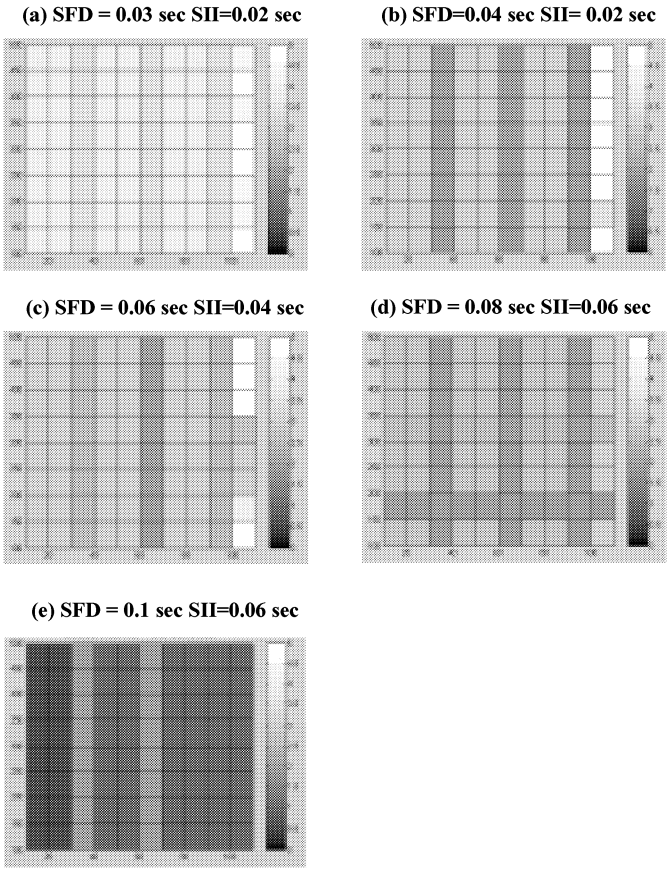


Figure 2: Simulation Effects at different SFD and SII

We may notice from the graphs that for gears at speed of 30 rps, 60 rps, 90 rps, the animation images are not good for all SFD and SII, jumpy and slow down when SFD is 0.03 seconds with SII at 0.02 seconds, and jumpy and backward moving at the other SFD. More sets of simulation find that starting at 2:00 o'clock or 3:00 o'clock instead of 12 o'clock position cannot solve the 30-60-90 problems. When SFD is kept constant, setting SII at 0.0617, 0.0623, 0.0655 seconds found the images are still jumpy and backward moving. Refinement of the simulation shows that immediately above or below, i.e. 31-61-91 rps and 29-59-89 rps also have the jumpy and backward moving effects as the 30-60-90 rps. Animation effects do not make much difference for 30, 60 or 90 rps when the refreshment rate is changed from 75 Hertz to 85 Hertz. We have not yet been able to find an explanation for the 30-60-90 syndrome.

On a final note, we should add that while the visual perception of the images depends on the observer, our survey on the perception of these images has found that there is reasonable overlap and consensus between the perceived observations.

4. CONCLUSIONS AND FUTURE WORK

This paper presents a series of problems for truthfully time scaled visualizations in modeling and simulation, and introduces the stroboscopic simulation that enables observation of VR objects moving simultaneously at very fast and slow speeds, maintaining their relative speeds truthfully. Detailed analysis of combinatorial simulation results was made to find the range in which stroboscopic simulation works well, and for which speed proportions it is not necessary. The durations and intervals of stroboscopic flashes that can provide suitable and pleasant dynamic images were also determined. Future research is aimed at stroboscopic illumination to other typical dynamic structures of MEMS components, such as flicking cantilevers, pumping membranes, and combinations of flicking and rotating movements.

5. REFERENCES

Gepshtein, S. and Kubovy, M. (2001). "The Weights of Space and Time in the Perception of Visual Motion". *Journal of Vision*, Volume 1, No.3, the Association for Research in Vision and Ophthalmology (ARVO), Rockville, MD.

Goldstein, E. Bruce. 1996. *Sensation and Perception.* Brooks.Cole Pub., Pacific Grove, CA.

Kihm, K. D.; T.K. Kim and S.Y.Son. 1999. "Visualization of high-speed gas jets and their airblast sprays of cross-injected liquid". *Experiments in Fluids* 27, Springer-Verlag, Heidelberg, 102-106.

Li, Z. and R.Sitte. 2001."Virtual Reality Modelling Aiding in MEMS Design". In *Proceedings* of *Electronics and Structures for MEMS*(Adelaide, Dec.17-19).SPIE, Bellingham,WA,152-162.

Li, Z. and R. Sitte. 2003. "Modeling Unsynchronized Motion Dynamics for Truthful VR Visualizations*"*. In *Proceedings of the 2003 International Congress on Modeling and Simulation* (Townsvill, Australia, July 14-17).

Nance, Richard E. and Robert G. Sargent, 2002. "Perspectives on The Evolution of Simulation". *Operations Research*. Vol. 50, No. 1 (Jan.-Feb.). INFORMS, Linthicum, M.D., 161-172.

Schiffman, H. 1995. *Sensation and perception: An integrated approach.* Wiley, NY.

Sitte, R. 2003. "Introductory Physics Based Visual Simulation", In *Proceedings of the C# and .NET Technologies 2003*, Plzen (Czech Republic), V.Skala (Ed), ISBN 80-903100-3-6, 63-69.

Wertheimer, Gestalt. 1999. "Gestalt Theory Reconfigured: Max Wertheimer's Anticipation of Recent Development in Visual Neuroscience". *Perception* Volume 2, Pion Ltd, London, 5-15.

Zeki, S. 1992. "The Visual Image in Mind & Brain. *Scientific American*, *241*(3), Scientific American, Inc., 150-162.

INTEGRATION OF SYSTEM DYNAMICS MODELS AND GEOGRAPHIC INFORMATION SYSTEMS

Stefano Mazzoleni
Francesco Giannino
Marco Colandrea
Massimo Nicolazzo
Facoltà di Agraria
Università di Napoli Federico II
Portici (NA), Italy
E-mail: mazzolen@unina.it

Jonathan Massheder
Simulistics Ltd.
Edinburgh Technology Transfer Centre
King's Buildings
Edinburgh, United Kingdom, EH9 3JL
E-mail: jmassheder@simulistics.com

ABSTRACT

The problem of integration of temporal models and Geographic Information Systems (GIS) involves theoretical and technical aspects. Many GIS applications are advertised in terms of their modeling capability but such modeling tools are often unable to handle full integration of spatial and dynamic processes. This paper, however, presents new software, SimArc, able to interface a System Dynamics tool, Simile, and a GIS environment (E.S.R.I. ArcView). The tool enables users to link a model to input data from the GIS database, creating new GIS layers from any selected output variable of Simile model.

KEYWORDS

GIS, System Dynamics, modeling.

INTRODUCTION

The problem of integration between temporal models and Geographic Information Systems (GIS) involves theoretical (Karimi, and Houston 1996, Yates and Bishop, 1998) and technical aspects (Goodchild et all. 1993, 1996 and Huang and Jiang 2002). Scaling problems are often recognized as a complex issue to be faced. Many GIS software packages are advertised in terms of their modeling capability (ArcView Spatial Modeler Extension, Erdas IMAGINE Model Maker). However such modeling tools are often unable to handle a full integration of spatial and dynamic processes. In the field of ecological modeling several papers have been published discussing the importance of both spatial and temporal scales (Jarvis and Mc Naughton, 1986; van der Maarel, 1988; Falinski, 1988; Allen and Hoekstra, 1990; Grace, 1991; Fahrig, 1992; Levin, 1992).
The scaling problem and the issue of linking dynamic and spatial processes requires integrated logic and an appropriate software solution. In this context, this paper presents work done at integrating a System Dynamics environment able to model processes in time and a GIS. So the integration of static spatial information within a dynamic system is the key modeling issue addressed (Figure. 1).

SIMILE: A SYSTEM DYNAMIC SOFTWARE

Simile (www.simulistics.com) is a recent addition to the family of System Dynamics modelling environments. There are several system dynamic software tools (see Costanza 1998, 2001 for review) that allow the construction of models by graphical interfaces, thus not requiring the use of conventional programming languages. These tools are able to automatically generate executable programs. Well known packages are STELLA (www.hps-inc.com) with a wide application in the field of ecology and agriculture modeling, SIMULINK (www.mathworks.com) supporting tool for MATLAB, characterized by strong computational capacity and access to mathematical libraries, other products to be mentioned are POWERSIM (www.powersim.com) and ModelMaker (www.cherwell.com).
Compared to the above mentioned products, Simile presents very interesting enhanced capabilities well suited for ecological modelling work (Muetzelfeldt and Massheder, 2003), and it is able to compile the models in C++ and/or export models as library files (Figure 1)

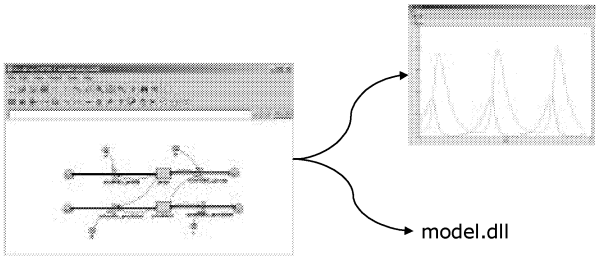


Figure 1: An example of graphical interface of a SIMILE model and types of outputs

SIMARC: SIMILE - ARCVIEW DYNAMIC LINK

The new software tool named SimArc (http://143.225.165.9/simarc/) was recently developed to interface models created by Simile with ArcView (E.S.R.I. Inc.) GIS environment. The tool enables users to link a model to input data from the GIS database. This essentially amounts to running a model in each polygon element of the ArcView map with inputs from layers of the relative shapefile and creating new GIS layers from any selected model output variable (Figure 2).

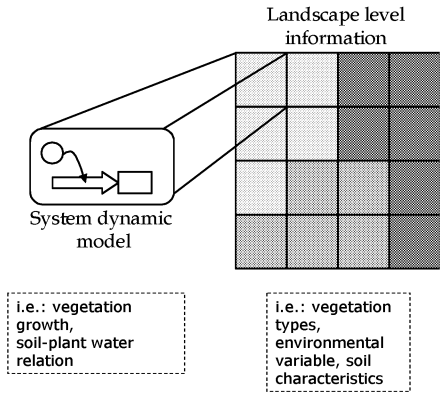


Figure 2: Upscaling to the landscape level in

In Microsoft Windows software can be linked by any of four different approaches (

Figure 3):

- Data file sharing
- Dynamic Data Exchange (DDE)
- Compiled model library (DLL)
- COM Automation (formerly known as OLE)

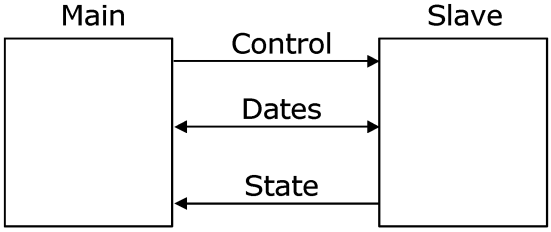


Figure 3: General software interaction

We used a mixed strategy to implement SimArc (Figure 4), in fact ArcView calls SimArc through a DDE protocol, and it calls the Simile's DLL to run the model in each polygon of the map. Finally, SimArc produces a new .dbf file to contain the output of the simulation and show as new layers in the ArcView map.

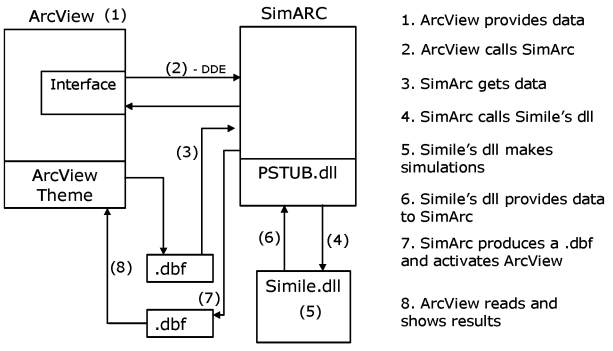


Figure 4: Data flow structure in SimArc

EXAMPLE OF INTEGRATED MODELLING ENVIRONMENT

Figure 5 shows a schematic representation of a simple SimArc application: a vegetation model is realized in Simile with just two input parameters from a GIS (temperature and water), and one output (biomass). SimArc links (by drag & drop commands) the input parameters with two ArcView maps to generate a new output map (vegetation biomass).

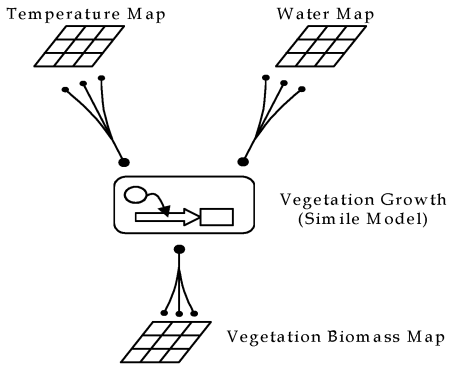


Figure 5: A schematic representation of a application from Simile and Arcview

Figure 6 shows a window of ArcView and the user interface of SimArc. In the SimArc window, on the left there is the list of all elements of Simile model, whereas on the right there are the layers information of .dbf ArcView file. It is possible to link an attribute of a layer in the ArcView map to an input element of the Simile model and, moreover, it is possible to define which variable of the Simile model of is the output as a layer of the map.

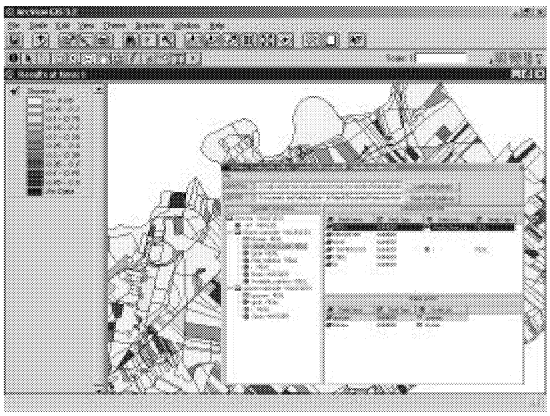


Figure 6: Screen shot of ArcView with the Simile extension. The SimArc window permits the connection of the Simile model to the ArcView map.

COMPUTATIONAL ISSUES

A Simile model with 32 elements (compartments, variables, flows) and a number of elements in ArcView of 40 polygons, required a running time less than 1 second for each time step (on a Intel Pentium IV 1GHz). The following table reports some running times for different case studies.

Table 1: Running Time of Different Simulations

Number of Simile elements	Number of ArcView polygons	Time (second)
32	40	0.8
32	~4000	1.3
32	~20000	3.1
81	~20000	5.3

CONCLUSION

SimArc links two powerful software packages, Simile (a System Dynamics modelling tool) and ArcView GIS, into an even more powerful synthesis.
Other modeling approachs consider the integration of spatial processes (Figure 7). To this aim we are developing a new, specific raster-based, spatial modelling system named 5D. The advantage of using this new, dedicated spatial modeling tool, instead of an established GIS package lies in its enhanced modelling capabilities, i.e. the possibility of making use of temporal simulations at both local and spatial scales in a highly integrated way. (A new Simile version enables integration with ArcGIS 8.0 by COM technology including use of Simile's inherent spatial capabilities.)

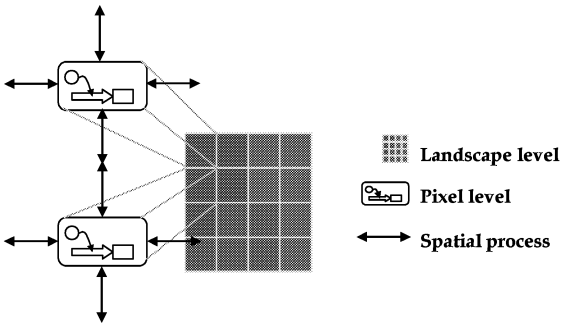


Figure 7: New software tools for integrated modeling require different System Dynamics models to interact, by different spatial processes, according to their position.

BIBLIOGRAPHY

Allen T.F.H. and Hoekstra T.W., 1990. The confusion between scale-defined levels and conventional levels of organization in ecology. J. Veg. Science, 1:5-12.

U. Berger, H. Hildenbrandt, A new approach to spatially explicit modelling of forest dynamics: spacing, ageing and neighbourhood competition of mangrove trees in Ecological Modelling (2000), 132, pp.287-302

Costanza, R., Gottilieb, S. 1998. Moddeling ecological and economic systems with STELLA: Part II. Ecological Modelling 112, 81-84.

Costanza, R., Voinov A. 2001. Moddeling ecological and economic systems with STELLA: Part III. Ecological Modelling 143, 1-7.

Fahrig L., 1992. Relative importance of spatial and temporal scales in patchy environment. Theoretical Population Biology, 41: 300-314.

Falinsky J.B., 1988. Succession, regeneration and fluctation in the Bialowieza Forest. Vegetatio, 77:115-128.

Goodchild, M.F., Parks, B.O., Steyaert, L.T. (Eds.), 1993. Environmental Modelling with GIS. Oxford University Press, New York(488 pp.).

Goodchild, M.F., Steyaert, L.T., Parks, B.O., Johnston, C.A., Maidment, D.R., Crane, M.F., Glendinning, S. (Eds.), 1996. GIS and Environmental Modeling: Progress and Research Issues. GIS World Books, Fort Collins, CO(504 pp.).

Grace J., 1991. Physical and ecological evaluation of heterogeneity. Functional Ecology, 5: 192-201.

Bo Huang and Bin Jiang 2002. "*AVTOP: a full integration of TOPMODEL into GIS*", Environmental Modelling & Software 17, 261–268

Jarvis P.G. and Mc Naughton K.G., 1986. Stomatal control of transpiration: scaling up from leaf to region. Advances in Ecological Research, 15:1-49.

Karimi, H.A., Houston, B.H., 1996. Evaluating strategies for integrating environmental models with GIS: current trends and future needs. Computers, Environment and Urban Systems 20 (6), 413–425.

Maarel E. van der, 1988. Vegetation dynamics: patterns in space and time. Vegetatio, 77: 7-19.

Mazzoleni, F. Rego, F. Giannino and C. Legg, Vegetation and disturbance in Wainwright, J. and Mulligan, M. (eds) (2003) Environmental modelling : a practical approach. John Wiley and Sons.

Mazzoleni, S. and Legg, C.J. (eds. 2001) ModMED: Modelling Mediterranean Ecosystem Dynamics, Final Report ModMED III Project, EU-DGXII Environment (IV) Framework, ENV4-CT97-0680.

Muetzelfeldt, R. and Massheder, J. (2003) The Simile visual modelling environment. Europ. J. Agronomy. (In press)

Yates, P. and I.D. Bishop. 1998. "*The integration of existing gis and modelling systems: with urban applications*" Comput., Environ. and Urban Systems, Vol. 22, No. 1, 71-80.

VIRTUAL REALITY APPLICATIONS

SPLINE APPROXIMATIONS OF FLEXIBLE DEFORMATIONS FOR FAST DYNAMIC VR VISUALIZATIONS

Kevin Tatur and Renate Sitte
Faculty of Engineering and Information Technology
Griffith University
Gold Coast, Australia
E-mail: kevintatur@hotmail.com

KEYWORDS
MEMS, VR, Fast Visualizations, CAD, FEM, Splines.

ABSTRACT

This paper presents the modeling of flexible structures in MEMS, using splines for fast calculations suitable for virtual reality visualizations in a CAD tool. In almost all MEMS devices thin films with thickness of a few microns or less, are applied to structure the components that frequently serve in essential device functions. For movable components, the demands placed on thin films in these applications can sometimes subject them to various mechanical conditions, such as; fracture, plasticity, friction and wear, creep, fatigue, etc. Most MEMS simulators show visualizations, which are static, while those showing dynamic visualizations, do so with FEM whose slow data processing is unsuitable for interactive VR. Despite increasing computational power and speed, most simulators are still far away from real time dynamics or rendering. Computations required for a visual simulation can take several hours requiring two phases: (a) calculations; (b) production of animated post-visualizations using the results of the calculations. This paper describes the use of splines as an effective means of finding shortcuts for faster physically truthful visualizations. By approximating the deflection profile of a circular MEMS membrane with three splines, we were able to model the sequence of movement of the deflection shapes with high accuracy.

1. INTRODUCTION

This paper presents a new method for the efficient modeling of bending objects using splines for fast calculations suitable for virtual reality (VR) visualizations in a CAD tool. Few Micro Electro Mechanical Systems (MEMS) simulators display animations of moving structures at all. Flexible deformations are typically calculated with Finite Elements Methods (FEM), whose slow data processing is unsuitable for interactive VR. This shortage can be overcome by building a MEMS simulator that displays VR visualizations as simulated functioning of the device being designed (Sitte 1999).

With the increase of CAD visualizations for industrial product development, a range of shortcuts has emerged for fast approximations of the visible details, for example surfaces (Tatur and Sitte 2003). However, more research is needed to fill the demand for shortcuts for faster physically truthful visualizations (Sitte 2003).

Many techniques and systems have been developed for object modeling and deformation. The most well known method for object deformation is *free-form deformation* (FFD). Sederberg and Parry first proposed the Free-Form deformation algorithms that change the geometric attributes of an object flexibility under some restrictions related to the properties of the object (Sederberg and Parry 1986). Since then, there have been substantial enhancements on the original FFD in the works of Coquillart, Lamousin, and Waggenspack (Coquillart 1990; Lamousin and Waggenspack 1994). Basically, all of them deform an object by deforming the space around it. The object is first embedded in or mapped to a 3D solid lattice defined by some parametric function. Deformation of the object can then be achieved by deforming the control points of the 3D lattice. However, deforming an object through manipulating control points is not intuitive to use.

Hsu et al. proposes a method that allows the user to move the sample points on the object model and automatically computes the necessary alteration to the control points of the FFDs (Hsu et al. 1992). However, this technique involves the computation of least squares, which is computationally very expensive. In addition, the problem can be very complex when more than one vertex point is to be moved at the same time, and the solution may not be unique. Similar to other FFDs, this technique only tackles the modification of control points one by one.

Recently, Piegl and Tiller presented a method to approximate surfaces of revolution with nonrational B-splines requiring a modest number of control points in the range of engineering tolerances (Piegl and Tiller 2003). They found that surfaces of revolution could be obtained by revolving a given curve around a specified axis at a given angle. These are often free-form and nonrational curves.

Figures 1a through 1d illustrate a practical design process. Figure 1a shows a set of points sampled from a conceptual drawing. Using an appropriate interpolation technique, the design intent was captured reasonably well. Once the profile was set, the system calls the revolution operator (extruding) and the surface shown in figure1c is obtained. Figure 1d shows the surface with the control points turned off.

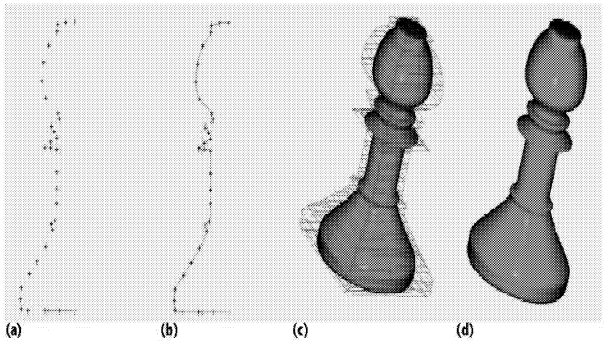


Figure 1: A Practical Design Process. (a) Data Points on the Profile Curve, (b) Profile Curve, (c) Surface of Revolution with Control Net, and (d) Surface of Revolution (Piegl and Tiller 2003).

However, this application of splines is an example of static VR visualization. In our case, we will be dealing with interactive dynamic VR visualizations, where not just one solution needs to be found, but a whole family of curves that describe the deformation at different times. It is the purpose of this paper to investigate how this can be tackled in an efficient way, suitable for interactive VR. While the application of splines is not new in visualizations, their application for the fast generation of physically truthful dynamics modeling is.

This paper is organized in the following way: Section 2 analyses the specific problem and constraints of the modeling of flexible parts in MEMS. Section 3 analyses the modeling and simulation issues for moving MEMS structures, in particular for membranes.

2. PROBLEM ANALYSIS

A void exists in the MEMS market where high-temperature applications (exceeding 1000^{o} C) are involved (Liew et al. 2002). Two categories of examples are: (1) MEMS that can operate in high-temperature environments, such as sensors for gas turbine engines, and (2) MEMS that can contain high-temperatures, such as micro mirrors for high-energy laser systems, and micro power generation systems. The latter will become especially important as maturing MEMS enter the market and increase the demand for completely remote, self-sufficient systems. Already numerous applications in defense and telecommunications do exist.

For these purposes, traditional MEMS are limited by their materials because silicon and polysilicon cannot withstand temperatures greater than several hundred degrees Celsius. Ceramics, mainly SiCN, present an attractive alternative to traditional MEMS for high- temperature applications.

3. MODELING AND SIMULATION

Solids are in general modeled with the particle approach. The total energy of their particles is rather low, the particles do not move freely. The preferred model for visualizations of solids is the spring model, derived from the Lennard-Jones model, as a special case approximation. For our purpose however, we are more interested with the truthful shape of the object bending and shaping under the influence of a force. In our experiment we use Finite Element Analysis to simulate the deformation of a circular membrane. We then fit a second order polynomial and derive a spline model to predict the shape of the bending device. Simulation is an important step in the design of MEMS, and the use of CAD tools help in reducing the overall costs and time between conception and prototyping up to 80%.

3.1 Membrane Modeling

The circular SiCN membrane was modeled and calibrated with ANSYS. The circular membrane shown in figure 2 was modelled and its edges were fixed to simulate the supports. In order to account for the circular geometry of the membrane, finite element analyses (FEA) were performed using ANSYS to obtain the membrane stiffness. The structure was meshed with SHELL63 elements. A Young's modulus of 150 GPa and a Poisson's ratio of 0.17 were used as input parameters accounting for material properties. SHELL63 elements were used since it is an elastic shell which has both bending and membrane capabilities. Both in-plane and normal loads are permitted. The element has six degrees of freedom at each node: translations in the nodal x, y, and z directions and rotations about the nodal x, y, and z axes. Stress stiffening and large deflection capabilities are included. A consistent tangent stiffness matrix option is available for use in large deflection (finite rotation) analyses.

We modeled the circular membrane as in figure 2, but with different dimensions to suit our experiments. Nine membranes were modeled in ANSYS with same diameter, but different thicknesses as listed in Table 1.

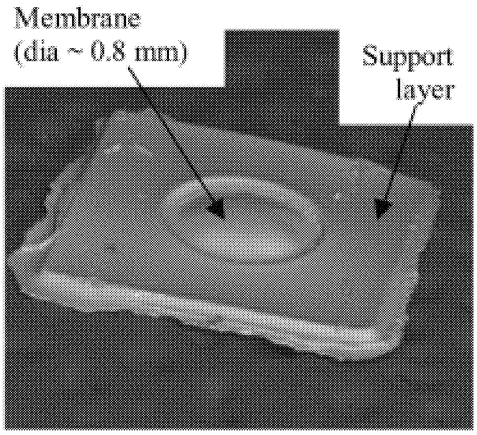


Figure 2: SEM of Pressure Transducer without Substrate, upside down. The Membrane is about 35 μm Thick (Liew et al. 2002).

Table 1: SiCN Membrane Dimensions

SAMPLE	DIAMETER (mm)	THICKNESS (μm)
1	0.2	10
2	0.2	5
3	0.2	2.5
4	0.2	8
5	0.2	4
6	0.2	2
7	0.2	6
8	0.2	3
9	0.2	1.5

We modeled all the membranes in 3D as cylindrical blocks to obtain a circular geometry, with the height of the cylindrical blocks illustrating the membrane thickness. Figure 3 illustrates the Isometric view of the membrane as generated by ANSYS and Figure 4 shows the meshed structure.

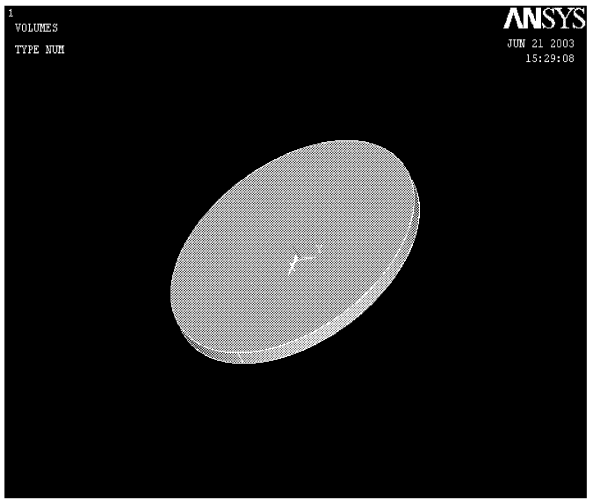


Figure 3: Isometric View of Sample 1

3.2 Simulation of Membrane Deformation

To simulate the membrane deformation, we used the transducer loading from the mechanical testing of (Liew et al. 2002). A micro-indentation was used to simulate transducer loading. This instrument accurately measures the position of a small probe while the load is varied. The probe was placed in the center of the membrane and the result of the deflection testing is shown in figure 5.

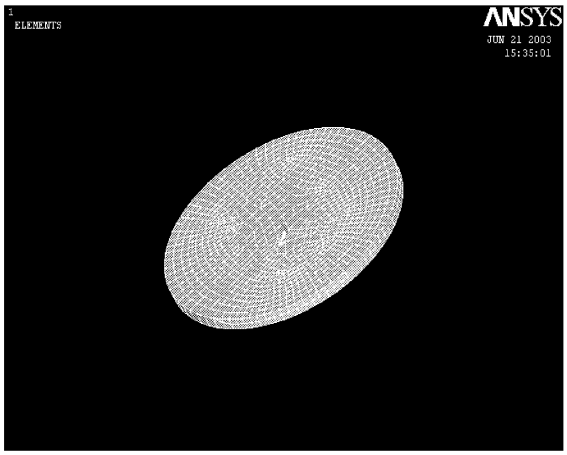


Figure 4: Sample 1 Meshed with SHELL63 Elements

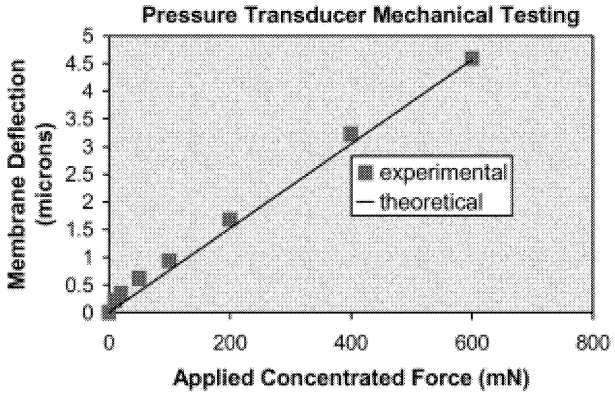


Figure 5: Results of Mechanical Testing (Liew et al. 2002).

The membrane was approximated as a circular plate with clamped edges, so that the deflection of the membrane as a function of an applied concentrated load is (Flugge 1962):

$$d = \frac{FR^2}{16\pi K}\left(\frac{2r^2}{R^2}\ln\frac{r}{R} - \frac{r^2}{R^2} + 1\right) \quad (1)$$

where d is the membrane deflection at a certain radial distance r from the center, R the radius of the membrane, F the applied concentrated load, and K is the flexural rigidity, which is proportional the membrane thickness to the cube and is given by:

$$K = \frac{Et^3}{12(1-\nu)} \quad (2)$$

where E is the Young's modulus of SiCN, ν is the Poissons ratio, and t is the membrane thickness.

Once the circular membranes were modeled and meshed in ANSYS, we used a loading of 100 mN, which is of the same order of magnitude as the mechanical testing shown in figure 5, to simulate the loading in ANSYS. The force was applied at the center of the membrane and the resulting deflection shape is illustrated in figures 6 and 7.

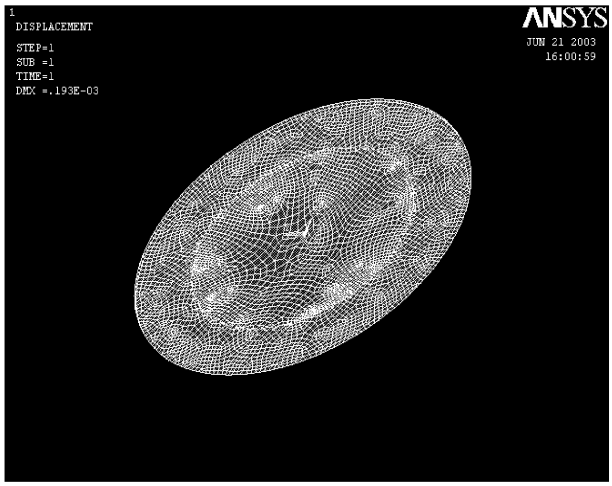


Figure 6: Deformed Shape of Sample 6

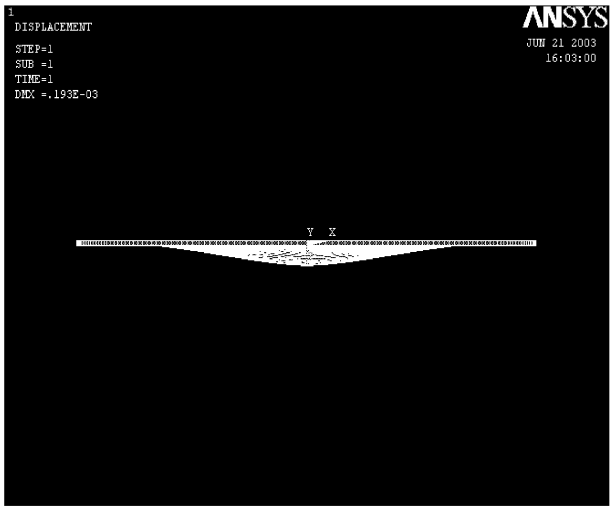


Figure 7: Side View Showing the Deflection Curve

4. SPLINE APPROXIMATIONS

In this section we describe the fitting process of the data points obtained from ANSYS. The data generated from ANSYS are for all the elements of the membrane. However, the ones of interest to us are those nodes found along the diameter of the membrane since the deflection profile is along the diameter.

For 2D screen visualizations, the information of the internal 3D structure of a body obtained with FEM is often not relevant. So we chose the surface nodes to get the deflection profile, and we approximated these data with splines and quadratics. First, a spline and a quadratic is fitted to all the data points as in (Tatur and Sitte 2003), and the errors and coefficients recorded. Then three splines and quadratics are used to describe the deflection profile of the membrane.

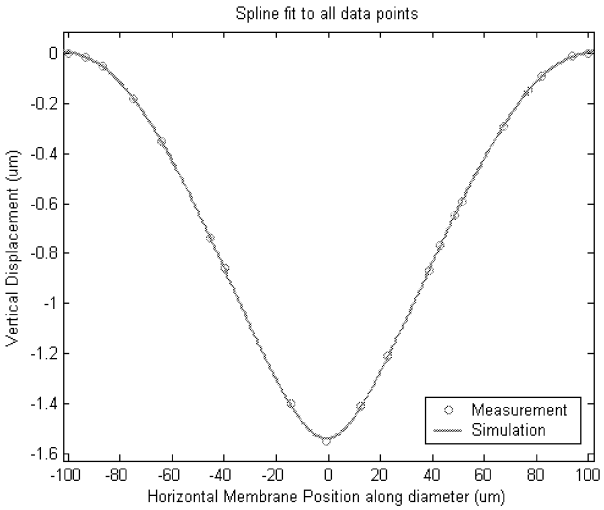


Figure 8: Smoothing Spline Fit to all Data Points for Sample 1

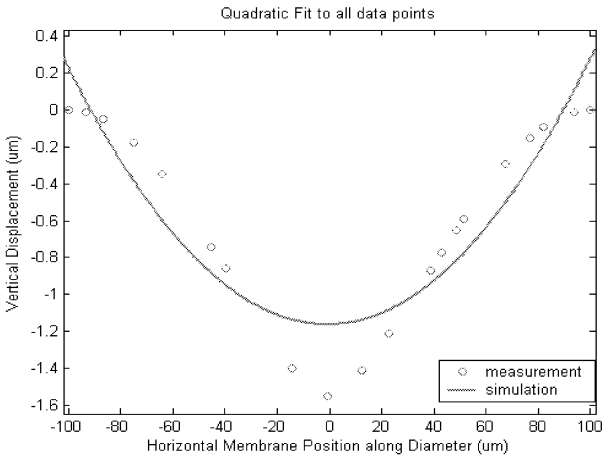


Figure 9: Quadratic Fit to all Data Points Sample 1

The following diagrams illustrate the deflection of sample 1 in three segments:

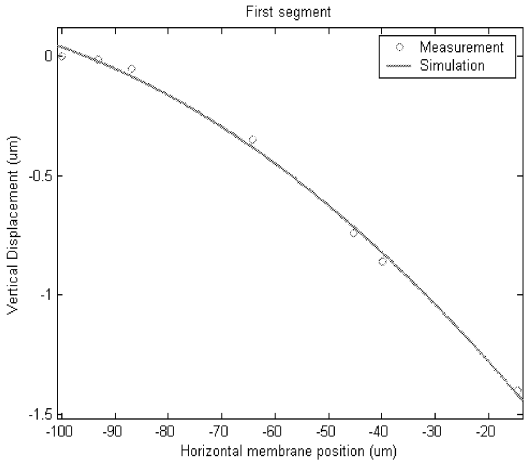


Figure 10: Quadratic Fit of First Segment

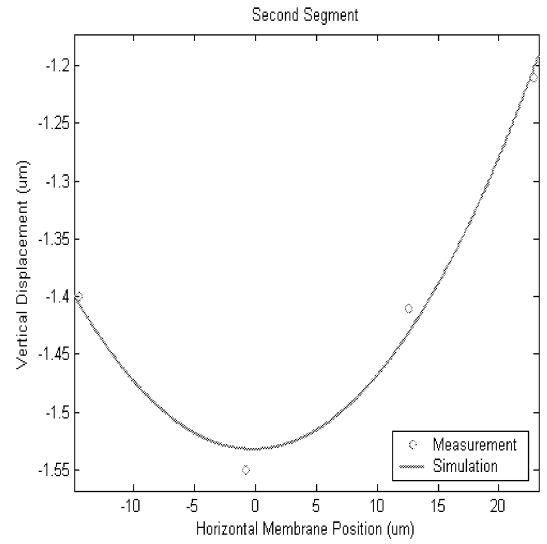


Figure 11: Quadratic Fit of Second Segment

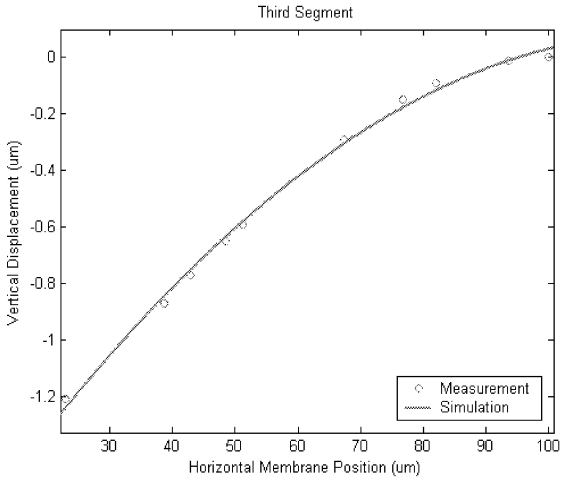


Figure 12: Quadratic Fit of Third Segment

5. MODEL DERIVATION

Our model makes use of Transition Matrices to predict the outcome of the deflection profile of other membranes. The Transition Matrices are obtained by halving the membrane thickness. For example, for the transition of thickness from Sample 1 to Sample 2, the Transition Matrix is obtained by dividing the deflection expression of Sample 2 by that of Sample 1. This Transition Matrix will be used to predict the deflection expression of Sample 3. This procedure is done for all the three segments of the Samples, i.e., we will predict the outcome of the segments rather than the outcome of the whole structure.

Consider the Thickness Transition of Sample 1 to Sample 2. For First Segment:

Equation of first segment of sample 1 (Figure 10):

$$Y11 = -0.00011x^2 - 0.029x - 1.83 \quad (3)$$

Equation of first segment of sample 2:

$$Y21 = -0.0013x^2 - 0.29x - 16.01 \quad (4)$$

The Transition Matrix in this case is:

$$T1 = Y21/Y11 \quad (5)$$

$$T1 = \begin{bmatrix} 0 & 0 & 0.0007 \\ 0 & 0 & 0.1585 \\ 0 & 0 & 8.7486 \end{bmatrix} \quad (6)$$

Predicted Equation of first segment of sample 3 is obtained by:

$$Y31' = T1xY21 \quad (7)$$

$$Y31' = -0.011x^2 - 2.54x - 140.07 \quad (8)$$

Where Y31' indicates the predicted outcome of the first segment of Sample 3.

Similarly, with the Equations of the second and third segments of Samples 1 and 2, we predict the outcome of the second and third segments of sample 3:

Predicted Equation of second segment of sample 3:

$$Y32' = 0.038x^2 + 0.0054x - 98.24 \quad (9)$$

Predicted Equation of third segment of sample 3:

$$Y33' = -0.0087x^2 + 2.049x - 120.51 \quad (10)$$

We have used the same procedure for the Thickness Transition of Sample 4 to Sample 5, to predict the segments of Sample 6. Also, We have predicted the segments of Sample 9 using the Transitions of Samples 7 and 8.

6. RESULTS AND DISCUSSIONS

In our previous work, (Tatur and Sitte 2003), we had fitted a complete rectangular cantilever with one single spline and quadratic and we got excellent results with small errors. However, here we are dealing with a circular membrane. We have chosen to predict the outcome of the segments rather than the outcome of the whole membrane to reduce the error margin of the simulation. The error margin for the simulation of the deflection of the whole membrane is of the order of 25% that is far too big.

The errors generated by the segments have a maximum of 0.22 for SSE and 0.39 for RMSE, which are very small and close to 0. This validates our reason for working with segments rather than the whole structure. The errors, SSE and RMSE, generated by the segments are small and better suited for the experiments as shown in the tables 2 and 3.

Table 2: Errors and Coefficients for Spline Fit of Sample 2

Segment	SPLINE FIT		
	Coefficient	Error	
		SSE	RMSE
1	0.012	0.0025	0.033
2	0.026	0.0042	0.054
3	0.022	0.0015	0.024

Table 3: Errors and Coefficients for Quadratic Fit of Sample 2

Segment	QUADRATIC FIT		
	Coefficients	Error	
		SSE	RMSE
1	p1 = -0.0013 p2 = -0.29 p3 = -16.01	0.13	0.16
2	p1 = 0.0047 p2 = 0.00067 p3 = -12.26	0.088	0.17
3	p1 = -0.0011 p2 = 0.26 p3 = -15.29	0.22	0.19

6.1 Predicted Models

The Predicted models obtained from section 5 are compared with the actual models obtained from ANSYS. The predicted segments of Sample 9 are shown in the following figures:

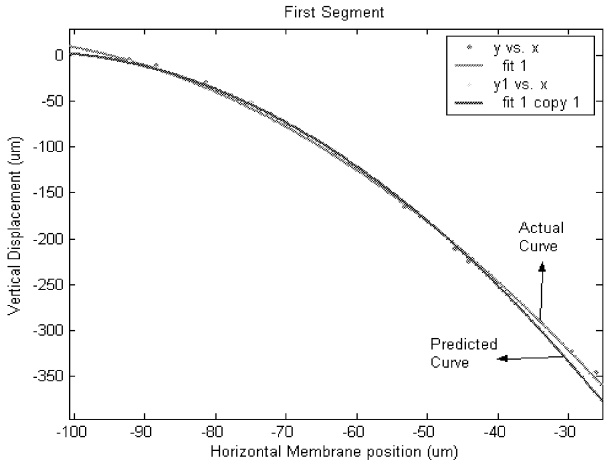


Figure 13: Predicted Curve and Actual Curve for first Segment of Sample 9

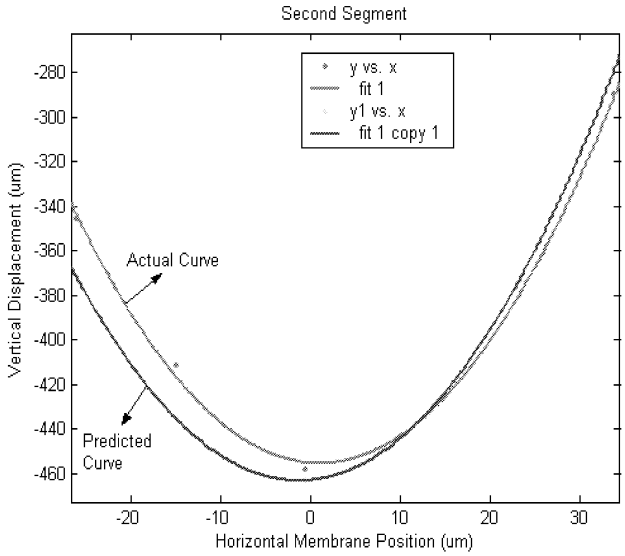


Figure 14: Predicted Curve and Actual Curve for second Segment of Sample 9

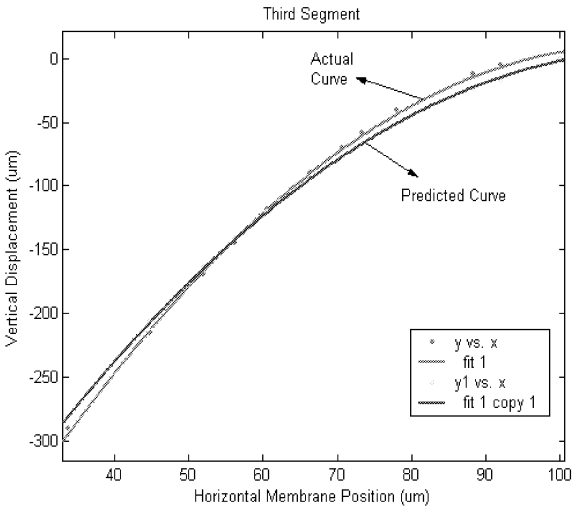


Figure 15: Predicted Curve and Actual Curve for third Segment of Sample 9

It can be seen from the figures that the predicted segments follow the actual segments very closely with little deviations. This is the same for the predicted segments of samples 3 and 6. The difference between the predicted curve and the actual curve stems from the fact that the data of the actual curve is obtained from ANSYS, which already has more than 10% error in its calculations. However, the overall deflection profiles of the segments are very similar to those of the actual deflection profiles.

6.2 Error Estimations

The sensitivity of our results deduced from the FEM may be affected up to 10%. This is the general error allowed by ANSYS and not necessary the error of the simulation.

The errors of our predictions are in the range of 1.4% to 15%, which are very good compared to the 25% error

generated by the whole segment and considering the fact that our results are affected up to 10% by ANSYS. The average errors of all the segments of a sample decrease as the membrane thickness decreases.

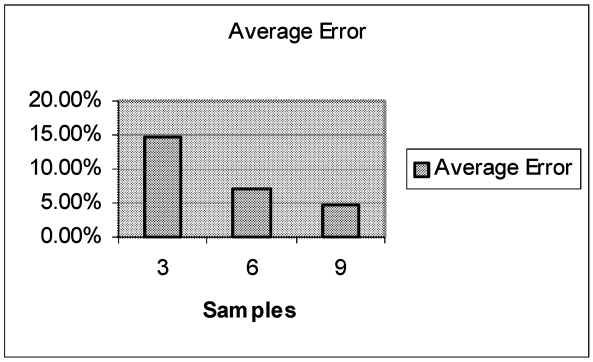


Figure 16: Average error of the predicted segments

Moreover, by approximating the whole deflected membrane by a single second order polynomial would underestimate the maximum deflection by around 30%. However, by approximating the deflection profile by three quadratics, it underestimates the maximum deflection of the membrane by only 0.97 %, which is a very good approximation.

7. CONCLUSION AND FUTURE WORK

We have presented a new method and technique for fast visualizations in VR using splines. We have described the modeling of MEMS flexibility using splines for fast calculations suitable for virtual reality (VR) visualizations in a CAD tool. We found that splines can be used as versatile and effective means of finding shortcuts for faster physically truthful dynamic visualizations. By approximating the deflection profile of a circular SiCN MEMS membrane with three splines, we were able to predict the deflection shapes of other membranes with a good degree of accuracy. We have approximated the deflection profile of the membrane with three polynomials rather than a single polynomial to reduce the error. We have demonstrated the use of Transition matrices to predict the shapes of segments and found that the predicted shapes follow the actual ones very closely with deviations ranging from 1.4% to 15%. ANSYS FEA, which has less than 10% error, has been used to calibrate our model. The experiments in this research have been conducted on nine samples, all with the same diameter but different thicknesses. Future research is aimed at refining the method to incorporate the diameter as a function parameter, and the material's Young's Modulus to model membranes with varying diameters and thicknesses, towards a generalization of the model. Future work also needs to include the force applied, and the time it takes to deflect to be used as input to the CAD visualizations.

8. REFERENCES

Coquillart S. 1990. "Extended Free-Form Deformation: A Sculpting Tool for 3D Geometric Modeling." *ACM Computer Graphics* 24, No.4, 187-193.

Flugge W. 1962. *Handbook of Engineering Mechanics*. 1st Edition, McGraw-Hill, New York.

Hsu W; J. Hughes; and H. Kaufmann 1992. "Direct Manipulation of Free-Form Deformation." *ACM Computer Graphics* 26, No2, 177-184.

Lamousin H. and W. Waggenspeck. 1994. "NURBS-based Free-Form Deformations." *IEEE Computer Graphics and Applications*, 59-65.

Liew L. A; Y. Liu; R. Luo; T. Cross; L. An; V. M. Bright; M. L. Dunn; J. W. Daily; and R. Raj 2002. "Fabrication of SiCN MEMS by photopolymerisation of pre-ceramic polymer." *Sensors and Actuators A* 95, 120-134.

Piegl L. and W. Tiller. 2003. "Approximating Surfaces of Revolution by Nonrational B-plines." *IEEE Computer Graphics and Applications* 23, No. 3 (May-June), 46-52.

Sederberg T. and S. Parry. 1986. "Free-Form Deformation of Solid Geometric Models." *ACM Computer Graphics* 20, No. 4, 151-160.

Sitte R. 2003, "Introductory Physics Based Visual Simulation", *Proceedings of the C# and .NET Technologies 2003,* Plzen (Czech Republic), V.Skala (Ed), ISBN 80-903100-3-6, 63-69

Sitte R. 1999, "Modeling MEMS Manufacturability with Virtual Prototyping CAD tools", *Electronics and Structures for MEMS II, Neil Bergman, Editor, Proceedings of SPIE* 4591, 125-133

Tatur K. and R. Sitte 2003. "Modeling Flexibility with Spline Approximations for Fast VR Visualizations", *Proceedings of the Modeling and Simulation Conference (MODSIM 2003)*, Townsville, July, **ACCEPTED**.

AUTHOR BIOGRAPHY

KEVIN TATUR was born in Vacoas, Mauritius and went to the University of Mauritius, where he studied Physics and obtained his BSC (Hons.) degree in Physics in 2001. He did his Masters by Research in Information Technology at the Griffith University. His area of interest is Computer Graphics and Visualizations.

RENATE SITTE obtained her PhD (1995) and MPhil (1993) from Griffith University in Microelectronic Engineering. She received her Systems Engineering degree (Ingeniero de Sistemas) in 1985 from the Universidad de Los Andes in Venezuela. She is a Senior Lecturer in the Faculty of Engineering and Information Technology, Griffith University, Australia, where she has been faculty staff since 1986. Her field of research is modeling and simulation, one major ongoing project is Virtual Reality Prototyping Micro-electro-mechanical systems (MEMS).

Automatic 3D object placement for 3D scene generation

Yoshiaki Akazawa[1], Yoshihiro Okada[1, 2] and Koichi Niijima[1]
[1]**Graduate School of Information Science and Electrical Engineering**
Kyushu University
6-1 Kasuga-koen, Kasuga, Fukuoka, 816-8580 JAPAN
{y-aka, okada, niijima}@i.kyushu-u.ac.jp
[2]***Intelligent Cooperation and Control, PRESTO, JST***

Keywords: Automatic placement, Semantic database, 3D scene construction, Virtual reality

Abstract
This paper describes an automatic 3D object placement system for 3D scene generation. The layout work for a 3D scene takes a lot of time because 3D objects have six degrees of freedom (DOF) and are usually controlled by a 2D input device. To deal with this problem, the authors propose a framework to automatically generate 3D scenes. The framework mainly consists of placement constraints represented as a semantic database of 3D objects. With this framework, the prototype system can automatically lay out most 3D objects without user manipulations. This paper mainly describes this framework, and delineates its usefulness by showing experimental results.

1. INTRODUCTION

This paper treats an automatic 3D object placement framework for 3D scene generation. The layout of 3D objects is an inevitable task for developing 3D graphics applications. However, it takes long time because a 3D object have six degrees of freedom (DOF) and is difficult to position using a 2D device, e.g., a mouse-device.

Many researches on 3D object layout have been made so far. Smith et al. [1] proposed the manipulation of 3D objects using a 2D user interface. This system employs contact constraints among 3D objects to allow the user to position 3D objects using a mouse-device. However, when laying out many 3D objects, it still takes long time even if using this system. Coyne and Sproat [2] proposed the WordsEye system, which generates 3D scenes composed of 3D objects according to text description. This method allows a few kinds of 3D scenes to be generated, based on a few lines of text. However, when laying out a large various 3D scene, the text description approach cannot lay out 3D objects accurately. Zeng el al. [3] also proposed natural language approach for 3D scene construction. For complex scenes, their system requests the user to write many sentences. Xu et al. [4] introduced an automatic placement system of 3D objects through user interaction. This system drastically reduces time consuming in 3D scene generation. However the system lays out 3D objects only on the floor of a room. It neither consider layout on a ceiling nor on a wall. In contrast, our system considers layout of 3D objects on the ceiling, as well as the floor by the same framework.

The remainder of this paper is organized as follows. Section 2 explains placement constraints. Section 3 explains object placement in detail. Section 4 introduces our prototype system and its placement results. Finally, Section 5 concludes the paper.

2. PLACEMENT CONSTRAINTS

2.1 Bounding box and occupancy space
When laying out 3D objects, each 3D object has to avoid colliding with other objects and each has contact constraints with the floor, the wall or the ceiling. It is difficult to calculate their positions satisfying such conditions because 3D objects have their own complex shape. To simplify the layout process, we decided to employ the bounding box of each 3D object instead of its original 3D shape as shown in Figure 1.

All of 3D objects have to consider distances among each other. For example, as shown in Figure 1, a bookshelf needs some space in its front in order to take a book out. Face 1 of a bookshelf has to be kept away from the faces of other objects. In this way, some faces of a 3D object have the minimum distance not to touch other objects. We call it “occupancy distance”.

Using the bounding box and the occupancy distance, a 3D object has its own space to avoid other objects. We call it “occupancy space”. The system can lay out 3D objects by collision detections based on occupancy spaces of existing 3D objects.

2.2 Parent-child relationship and contact constraint
Every object in the real world has to touch other objects because of the gravity. For example, face 4 of the bookshelf shown in Figure 1 must touch the face of a floor. If the user moves the floor object, the bookshelf should move with it. This is treated as the parent-child relationship in common among 3D applications. Every 3D object has information indicating what kind of 3D objects allow to be its parent, which face of the 3D

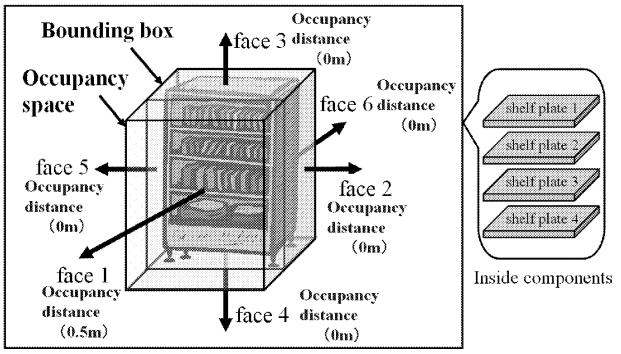


Figure 1. Bounding box, occupancy space and inside components of a bookshelf.

object and which face of its parent object touches each other. In addition to the above information, we have to specify contact constraint for each face of a 3D object. This indicates that the corresponding face should touch the certain face of other object or not. For example, a bookshelf often touches a wall in addition to a floor. That is, face 6 of the bookshelf in Figure 1 has to touch a wall object.

2.3 Semantic database

The above placement constraints are treated as a semantic database. The system lays out 3D objects using the semantic database. Every 3D object belongs to any object type. Each object type is defined by the information such as shown in Table 1 separately. This information is called 'object info'. The semantic database consists of multiple object info. In the object info, the distance attribute means occupancy distance for each face. Every object type has its parent object types. Only one of six faces is specified as the face that touches the parent. This is specified in the parent attribute of that face. The other faces of a 3D object has contact constraints specified in the contact constraint attribute.

2.4 Inside placement

Our system considers occupancy spaces of 3D objects to avoid their collisions. However, some objects exist in the bounding box of other 3D object. For example, as shown in Figure 1, books are placed on the shelves of the bookshelf. For this case, we also assign a bounding box to each shelf of the bookshelf manually and add 'object info' of a shelf into the semantic database. The object type of shelves should be indicated in the inside components attribute of the bookshelf object info in order to allow child objects of a shelf to exist inside of the bookshelf.

3. OBJECT PLACEMENT

3.1 Placement procedure

The system lays out 3D objects using the semantic database. At first, the system generates a parent-child

Table 1. Semantic database

ot_4(bookshelf)

face	occupancy distance	Parent	contact constraint	Inside components
1	0.5	-	-	ot_6
2	0	-	-	
3	0	-	-	
4	0	ot_1	×	
5	0	-	-	
6	0	-	ot_2	

ot_5(TV)

face	occupancy distance	Parent	contact constraint	Inside components
1	0.5	-	-	—
2	0	-	-	
3	0	-	-	
4	0	ot_4-3, ot_7-3	×	
5	0	-	-	
6	0	-	-	

OT(Object Type): {ot_1, ot_2, …, ot_n}

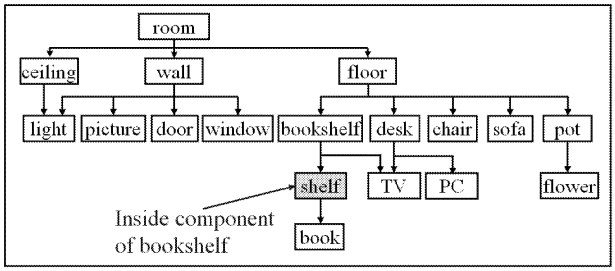


Figure 2. Parent-child relationship graph of the semantic database.

relationship graph from the semantic database as shown in Figure 2. Then, the system places each 3D object randomly based on the parent-child relationship using this graph. After that, the system rotates and moves the 3D object to the position that satisfies the contact constraints written in the semantic database. The system repeats the random layout several times until the result satisfies the user. The concrete placement algorithm is as follows.

1. Select one object type according to the distance from the root object type in the parent-child relationship graph. This distance is the priority of the selection. If there are multiple object types in the shortest distance from the root object type, select one object type that has the maximum number of contact constraints.
2. Choose one 3D object, which belongs to the selected object type, from the 3D objects the user prepared.

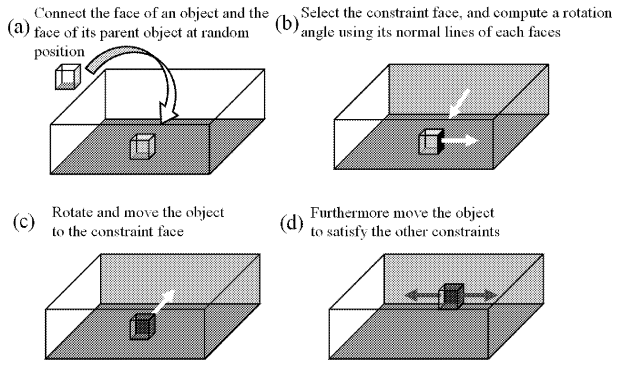


Figure 3. Four steps of the placement of an object having two contact constraints.

3. Randomly choose another 3D object as the parent of the 3D object chosen in Step 2 according to the graph.
4. Place the object chosen in Step 2 at random position on the object chosen in Step 3 as shown in Figure 3 (a).
5. Rotate and move the object to the position that satisfies all contact constraints as shown in Figure 3 (b, c, d) according to the semantic database record.
6. This random placement in Steps 4 and 5 is repeated until the object does not penetrate into other objects.
7. The Steps 3-6 will be applied to all 3D objects that belong to the object type selected in Step 1.
8. The Steps 2-7 will be applied to all object types in the graph.
9. Repeat the above steps until the placement result satisfies the user.

3.2 Group placement

As explained in previous subsection, the system places 3D objects using the random layout method. However, the random layout is not enough for generating realistic scenes because in the real world, objects of same type are often located together or located by strict layout rules. For example, desks in a class room are laid out by a strict layout rule. For such cases, we have to prepare some templates of typical layouts. Although this mechanism is not implemented, the system will be able to lay out 3D objects according to the user-selected template.

4. PROTOTYPE SYSTEM

A prototype system is developed using *IntelligentBox* [5, 6], which is a constructive visual 3D software development system. Figure 4 shows four placement results generated by the system. The 3D objects of each result are laid out like those in the real world. The user selects desirable one out of the results. After the user obtains his/her desirable layout, he/she can interactively move and rotate any 3D object according to the contact constraints specified in the semantic database. As for the performance, the execution time is around a few seconds when the number of objects is around 40.

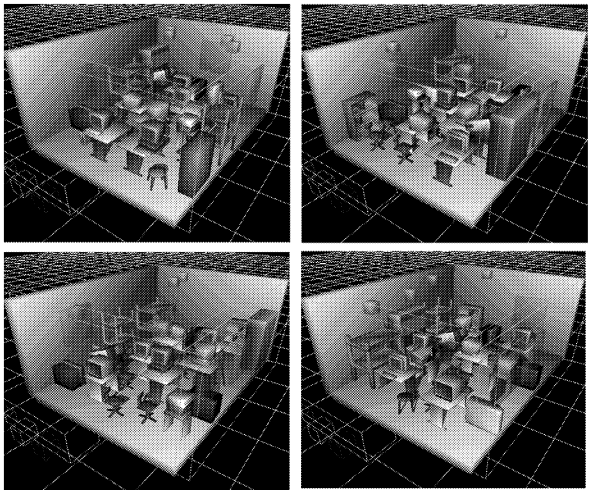


Figure 4. Four results of random layouts generated by the prototype system.

5. CONCLUSION

This paper proposed an automatic 3D object placement system for 3D scene generation. We introduced the semantic database, based on the bounding boxes of 3D objects, that defines contact constraints among the faces of bounding boxes. With the proposed placement method using the semantic database, the system automatically generates 3D scenes even if there are huge 3D objects in the scenes. As the future work, we will develop GUI to incrementally enter new data record into the semantic database.

REFERANCES

[1] G. Smith, T. Salzman, and W. Stuerzlinger, "Integration of constraints into a VR Environment", VRIC 2001, pp. 103-110, ISBN 295157300-6, 2001

[2] B. Coyne and R. Sproat, WordsEye: an automatic text-to-scene conversion system. SIGGRAPH 2001, pp. 487-496, 2001

[3] Zeng, X., Mehdi, Q. H. and Gough, N. E. 2003 "Shape of the Story: Story Visualization Techniques" Proc. of Information Visualization 2003 (IV03), 144-149.

[4] K. Xu, A. J. Stewart, and E. Fiume, "Constraint-Based Automatic Placement for Scene Composition", Graphics Interface, pp. 25-34, 2002

[5] Y. Okada, and Y. Tanaka,: IntelligentBox: A Constructive Visual Software Development System for Interactive 3D Graphic Applications, Proc. of Computer Animation '95, IEEE Computer Society Press, pp.114-125,1995.

[6] Y. Okada, and Y. Tanaka,: Collaborative Environments of IntelligentBox for Distributed 3D Graphics Applications, The Visual Computer, Vol. 14, No. 4, pp. 140-152, 1998.

SIMULATION IN BIOLOGY

SIMULATION IN BIOLOGICAL SYSTEMS

COMPUTER-AIDED PERFORMANCE ASSESSMENT OF A NEW MEASUREMENT ALGORITHM FOR ULTRASONIC-BASED SENSORS

Leopoldo Angrisani
Dipartimento di Informatica e Sistemistica
Università degli Studi di Napoli "Federico II"
Via Claudio 21, 80125 Napoli, Italy
E-mail: angrisan@unina.it

Rosario Schiano Lo Moriello
Dipartimento di Ingegneria Elettrica
Università degli Studi di Napoli "Federico II"
Via Claudio 21, 80125 Napoli, Italy
E-mail: rschiano@unina.it

KEYWORDS
Digital signal-processing algorithm, simulation-assisted performance assessment, time-of-flight estimation, statistical experimental design.

ABSTRACT
In order to assess the performance of a new measurement algorithm, a fundamental step is the analysis of its results when applied to numerical signals similar to actual ones. This way, the metrological characteristics of the algorithm could be analysed without the interference of the external uncertainty sources. This paper aims at describing the performance assessment of a digital signal-processing algorithm for ultrasonic time-of-flight (TOF) estimation. This algorithm will be implemented on an ultrasonic-based digital level smart-sensor. A suitable computer-aided technique of statistical experimental design has been used in order to reduce the simulation burden without loss in analysis significance. Numerical results has proven a satisfactory accuracy of the digital signal-processing algorithm.

INTRODUCTION

Ultrasonic-based measurement techniques are extensively used both in laboratories and industry, spanning in endless application fields such as *(i)* distance, level and flow measurements, *(ii)* non-destructive testing, and *(iii)* robot navigation [Sydenham and Thorn 1992]. A typical measurement technique exploits the estimation of the time-of-flight (TOF) of an ultrasonic pulse. One of the main advantages offered by this method is the possibility of building up relatively cheap meters with a satisfactory accuracy, and addressed to different industrial applications, such as level and distance measurements.
In these TOF–based measurements, an ultrasonic burst (i.e. a high-frequency sine pulse train) is generated from the transmitting transducer; the burst propagates through the transmission medium and is then reflected by any medium discontinuity or reflector. The returning pulse train is acquired by the receiving transducer (often the same transmitting one). The required distance or level is then achieved by estimating the time elapsed between the firing-up of the transmitted ultrasonic signal and the receiving of the echo, according to the following relationship:

$$d = \frac{c \cdot t}{2} \qquad (1)$$

where d is the measured distance, c is the propagation velocity of the ultrasonic wave, and, finally, t is the TOF estimation of the signal. The main sources of inaccuracy are [Eren 1995, Cai and Regtien 1993]:

1. uncertainty in TOF measurement;
2. distortion of the shape of the ultrasonic signal;
3. dependence on temperature of the sound propagation velocity in the transmission medium.

In the following sections, the paper describes the main features of a digital signal-processing algorithm for TOF estimation that allows the first of the aforementioned problems to be faced. This algorithm is able to assure accurate results by applying a classical I/Q coherent demodulation scheme to the acquired ultrasonic signal. This way, the right onset (i.e. the ultrasonic echo beginning) of the received echo can be identified (Fig. 1).
Performance assessment of the proposed digital signal-processing algorithm has been carried out through suitable numerical simulations. A set of synthetic numerical signals, characterised by known onset location, assumed as reference, allows the behaviour of the proposed algorithm to be verified avoiding the influence of the uncertainty introduced by the data acquisition unit and the external environment. In particular, bias, δ, (i.e. difference between the nominal and the estimated value of the TOF) and experimental standard deviation, σ, of the obtained TOF estimates were analysed upon the main signal parameters' varying in defined intervals.
Another important point to be considered is the efficiency of the analysis. To this end, a statistical experimental design-based technique was used in order to reduce the test burden without loss in analysis significance.

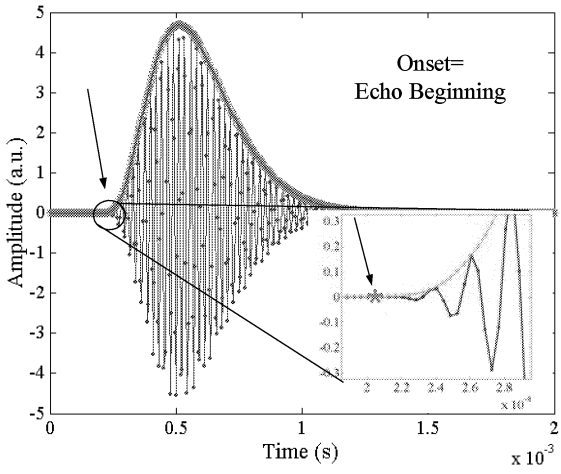


Fig. 1 – Onset location in TOF measurement.

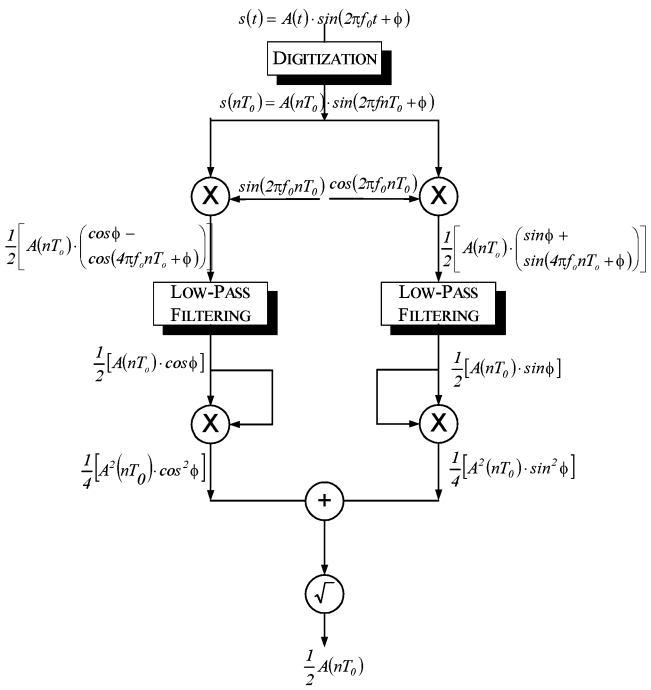


Fig. 2 – I/Q coherent demodulation scheme.

THE PROPOSED ALGORITHM

The fundamental steps of the proposed digital signal-processing algorithm are described in the following.

- *SIGNAL DEMODULATION*

  In the literature, the echo shape is usually modelled as:

  $$s(t)=A(t-\tau)sin(2\pi f_0 t+\phi), \qquad (2)$$

  where τ is the ultrasonic TOF, f_0 is the ultrasonic carrier frequency, and ϕ is a constant phase term. A suitable scheme, generally adopted for coherent I/Q demodulation of phase-modulated (PM) signals, has been then applied in order to recover the envelope $A(t-\tau)$ (Fig. 2). Two branches can be distinguished, the I branch and Q branch.

  With regard to the I branch, the acquired signal is multiplied by a discrete-time sinusoidal carrier, the frequency of which is equal to the nominal frequency of the input signal. The multiplication gives rise to both a base-band component and a high-frequency component. A low-pass filtering is carried out in order to retain only the base-band component (also called the I component). As for the I branch, the orthogonal version (also called the Q component) of the aforementioned base-band component is furnished as a result of similar operations. The filtered signals are subsequently squared and added in order to remove the dependence on the phase constant, ϕ.

  The low-pass filter has been generated by means of the Remez exchange algorithm, exhibits a linear phase behaviour, and shows moderate equiripple in the passband [Bishop 1998 - Oppenheim and Shafer 1992]. The parameters of the filter have been chosen in such a way that it has a very narrow bandwidth (lower than 1% of the sample rate) and a high stopband attenuation (greater than 80 dB).

- *LOCATION OF ECHO ONSET.*

  This step consists of two stages. In the first one, the position of the maximum of the echo envelope is determined. The onset is then associated to the first sample of the envelope in correspondence of which the derivative of the envelope itself change its sign. This result is achieved by running the rising side of the echo envelope for decreasing values of the time and evaluating the sign of the product of two consecutive derivative terms (Fig. 3).

STATISTICAL EXPERIMENT DESIGN-BASED NUMERICAL SIMULATION.

The digital signal-processing algorithm has been characterized through numerical simulations. In particular, for a defined range of the TOF, the effect on the final results of some measurement and algorithm parameters, such as shape factors and SNR of the signal, and carrier frequency in the demodulation step has been investigated. Use of numerically generated signal allows the performance of the only algorithm to be suitably assessed without the contribute of the other possible uncertainty sources.

Signal generation

A well-known [Sabatini 1997] expression has been used in order to suitably model the echo envelope:

$$A(t) = \left(\frac{t}{T}\right)^{\alpha} exp\left(\frac{t}{T}\right), \qquad (3)$$

where α and T are peculiar to the specific ultrasonic transducer.

The synthetic ideal signal of the echo envelope has been simulated as:

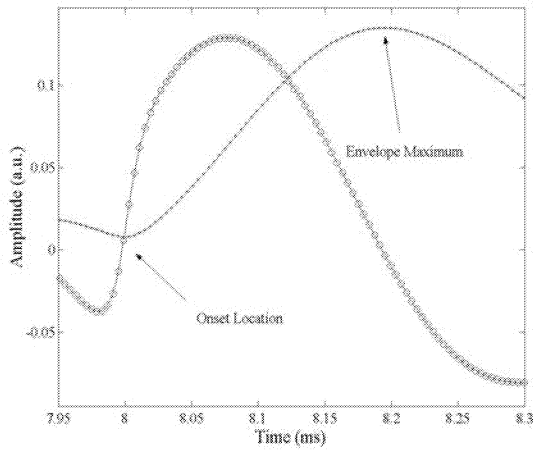


Fig. 3 – Once detected the envelope peak (red dotted line), the onset is located by searching the first sign change of the envelope derivative (blu circled line) before the maximum

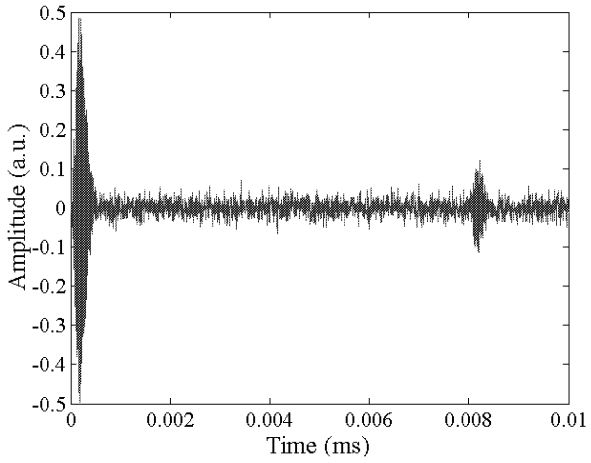


Fig. 4 – Example of synthetic ideal signal.

$$A(nT_0)=\left(\frac{nT_0}{T}\right)^{\alpha} exp\left(-\frac{nT_0}{T}\right)+ \quad , \qquad (4)$$

$$+\left(\frac{(n-m)T_0}{T}\right)^{\alpha} exp\left(-\frac{(n-m)T_0}{T}\right)U\left((n-m)T_0\right)$$

where T_0 is the sampling period, n is the generic sample, m is the sample corresponding to the ideal echo onset, and $U((n-m)T_0)$ is the m-delayed unitary step function.

The signal $A(nT_0)$ is subsequently *(i)* multiplied by the sinusoidal carrier whose frequency is equal to the ultrasonic one, *(ii)* multiplied by the function $exp\left(-\frac{nT_0}{T_1}\right)$ in order to simulate the effect of amplitude attenuation of the propagating ultrasonic pulse train, and *(iii)* added to a zero-mean white Gaussian noise (Fig. 4). The noise is the sum of two contributions, each of which is characterized both by zero mean and normal probability density function. The first contribution is band-pass (BW=3 kHz) in order to simulate the noise introduced by typical ultrasonic probe as well as connected equipment. The noise of the second contribution is white to properly model the quantization noise produced by the digitisation process.

Adopted procedure

Simulations based on a statistical experimental design have been performed, in order to evaluate the effect of the aforementioned parameters on the TOF estimation. According to the p-diagram description of the system [Montgomery 1997], the following parameters are considered: *(i)* as *input space parameter*, the TOF to be measured, *(ii)* as *configuration parameter (CP)*, the frequency $f_{I/Q}$ used for I/Q demodulation, *(iii)* as *error source parameters (EP),* the shape factors α and T, the signal-to-noise ratio, *SNR*, and the constant phase term, ϕ, and *(iv)* as *metrological performance*, the bias δ and the experimental standard deviation, σ, both expressed in relative terms.

The test aims at investigating the performance of the algorithm in the whole input space by evaluating δ and σ in the five-dimensional hyper-space determined by the CP and EP parameters. For each parameter, a suitable range of values, according to the actual intervals of their variation [Sabatini 1995], has been defined. In particular, the following ranges has been chosen:

1. TOF: 2-8 ms (corresponding to a distances range of 346-1384 mm at a temperature of 25 °C);
2. $f_{I/Q}$: 43.81-43.91 kHz (corresponding to a variation of ±50 Hz around the ultrasonic carrier frequency f_0);
3. *SNR*: 10-30 dB;
4. α: 1-4;
5. T: 58-98 μs;
6. ϕ: $\pm\pi/6$.

The experimental burden has preliminary been reduced by discretizing each variation range of the CP and EP parameters in 5 equally spaced levels (according to well-known statistical techniques [Montgomery 1997]), while the TOF interval has been discretized in 20 levels. For the sake of clarity, the discretized intervals of the CP and EP parameters are summarized in Tab.I.

Under the assumption of statistical independence of the 5 CP and EP parameters, a linear model can be assumed for δ and σ, and the combinatorial experimental test space can consequently be decimated by a suitable statistical experiment planning [Montgomery 1997]. This way, the whole number of tests, equal to 5^5, was reduced by selecting a standard fractional factorial Resolution III plan L25 for its capability of *(i)* exploring up to 6 five-level statistically independent parameters, and *(ii)* providing a repetition number n_t of 5. This allows the aforementioned test space of 5^5 experiments to be equivalently mapped by only 25 experiments, provided that suitable ANOM (analysis of mean) and ANOVA (analysis of variance) statistical processing techniques are applied, [Montgomery 1997].

Furthermore, for each experimental configuration of the aforementioned L25 plan (i.e. for each row of the matrix in Fig. 5), the space of the input parameter (*TOF*) have been mapped by executing, for each one of the 20 levels, 30 TOF estimations. These values have statistically been processed in order to evaluate δ and σ. Among this set of 20 couple of performance indicators, worst cases both for δ and σ have been considered as representative of the experimental configuration. Once executed the previous procedure for the whole L25 plan, ANOM and ANOVA analysis is conducted in order to estabilish the parameters really influencing the performance of the algorithm.

From an operating point of view, the adopted procedure consists of the following steps (Fig. 5).

1. A row of the L25 matrix is selected.
2. The algorithm is configured and the simulation signal is set by selecting the values of the parameters pointed

Tab. I – Discretised level for CP and EP parameters.

$f_{I/Q}$ (kHz)	43.81	43.84	43.86	43.89	43.91
SNR (dB)	10	15	20	25	30
α	1	2	2,5	3	4
T (μs)	58	68	78	88	98
ϕ (rad)	-0,524	-0,261	0,000	0,261	0,524

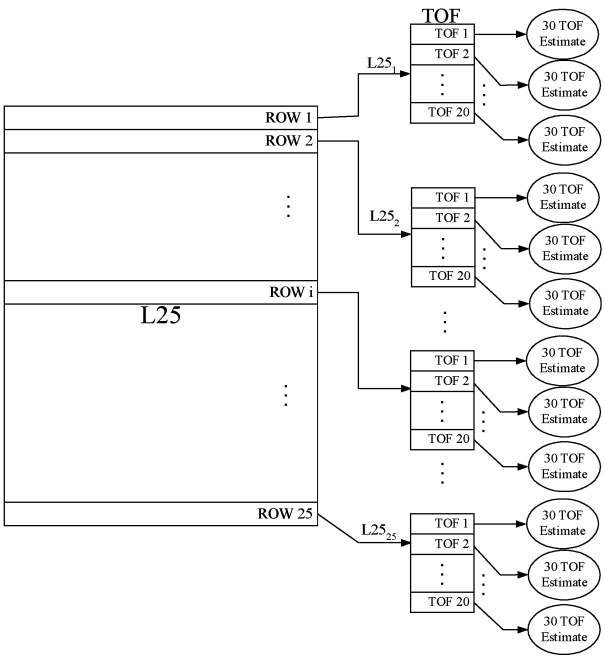


Fig. 5 – Basic scheme of the adopted design procedure.

out by the row.

2.1. A value of *TOF* among the 20 levels of the input parameter is selected.

2.2. 30 independent tests are carried out on signal generated each time adding a random noise, whose SNR is equal to the one specified in the row of the L25, to the ideal signal. This way, a set of 30 values of the TOF estimation is available.

2.3. δ and σ are obtained by means of statistical processing of the data.

2.4. If the input space has been completely mapped, the highest values of δ and σ are extracted as representative of the configuration, else the steps from 2.1 up to 2.4 are executed for the subsequent value of TOF.

3. If the whole L25 plan has been covered, a set of 25 couples of "worst" δ and σ are available (one for each algorithm configuration); else the steps from 1 up to 3 are executed for the subsequent row of the L25 plan.
4. On the set of 25 couples of δ and σ obtained at the previous steps, ANOM and ANOVA analysis is performed in order to evaluate the effects of each level of the CP and EP parameters on δ and σ.
5. The assumption of linear models is then validated by comparing their predictions with the corresponding simulation test results, and by assessing the prediction errors through statistical tests [Montgomery 1997].

As an example, results of the ANOM applied to the bias are shown in Fig. 6. The parameter level generating an effect greater than the dashed threshold influences significantly the bias with a confidence level of 95%.

A more quantitative significance analysis is reported in Tab. II (ANOVA): the ratio, *F*, between the variance imposed by the parameter, and the variance of the model error expresses the capability of a parameter of imposing a

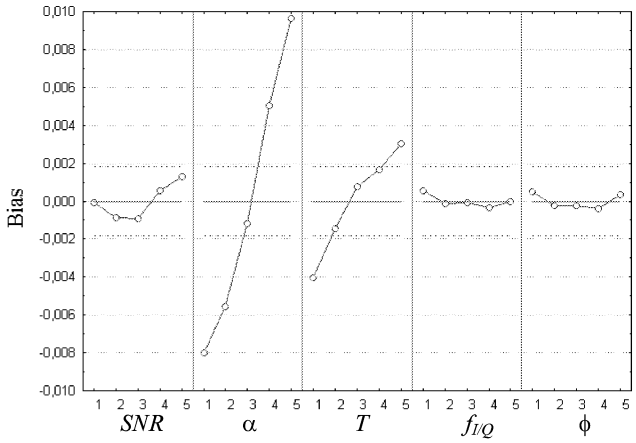


Fig. 6 – Results of ANOM for bias.

significant variation to the bias. This way, the shape factors have proven the most influencing parameters for the bias of the estimated TOF. In the same way, SNR shows to be the most influencing parameters for σ.

FURTHER TESTS

Once evaluated the significance of each one of the measurement parameters, an OFAT (One Factor At Time) analysis has been carried out for the two most significant parameters (SNR, α). Two different test planes ([TOF,SNR] and [TOF,shape factor]) has been produced to accurately investigate the effects on the algorithm performance (Tab. 3). In the first test plane, the value of the shape factor α has been fixed (α=3.5), and the surfaces of bias and σ, in relative terms, have been evaluated upon the TOF's and SNR's varying in the interval equal respectively to 2-8 ms and 10-40 dB. In the second test plane, an SNR value of 10 dB has been fixed and the surfaces have been evaluated upon the TOF's and shape factor's varying in the interval equal respectively to 2-8 ms and 3-4. The obtained results are shown in Fig. 7. From their analysis some considerations can be drawn:

- Values of bias and σ always smaller than 1% have been experienced.
- A satisfactory robustness to shape distortion has

Tab. II ANOVA of characterization results
(Par.: parameter, Fr.Deg.: Freedom degrees,
SS: sum of squares, F: variance ratio,
Sign.: confidence interval of significance)

	Bias			
Par.	**Fr. Deg.**	**SS**	**F**	**Sign.**
SNR	4	1,83E-05	1,09	0,533
α	4	1,07E-03	64,17	0,999
T	4	1,57E-04	9,37	0,974
$f_{I/Q}$	4	2,05E-06	0,12	0,033
ϕ	4	3,31E-06	0,20	0,073
Err	4	1,67E-05		

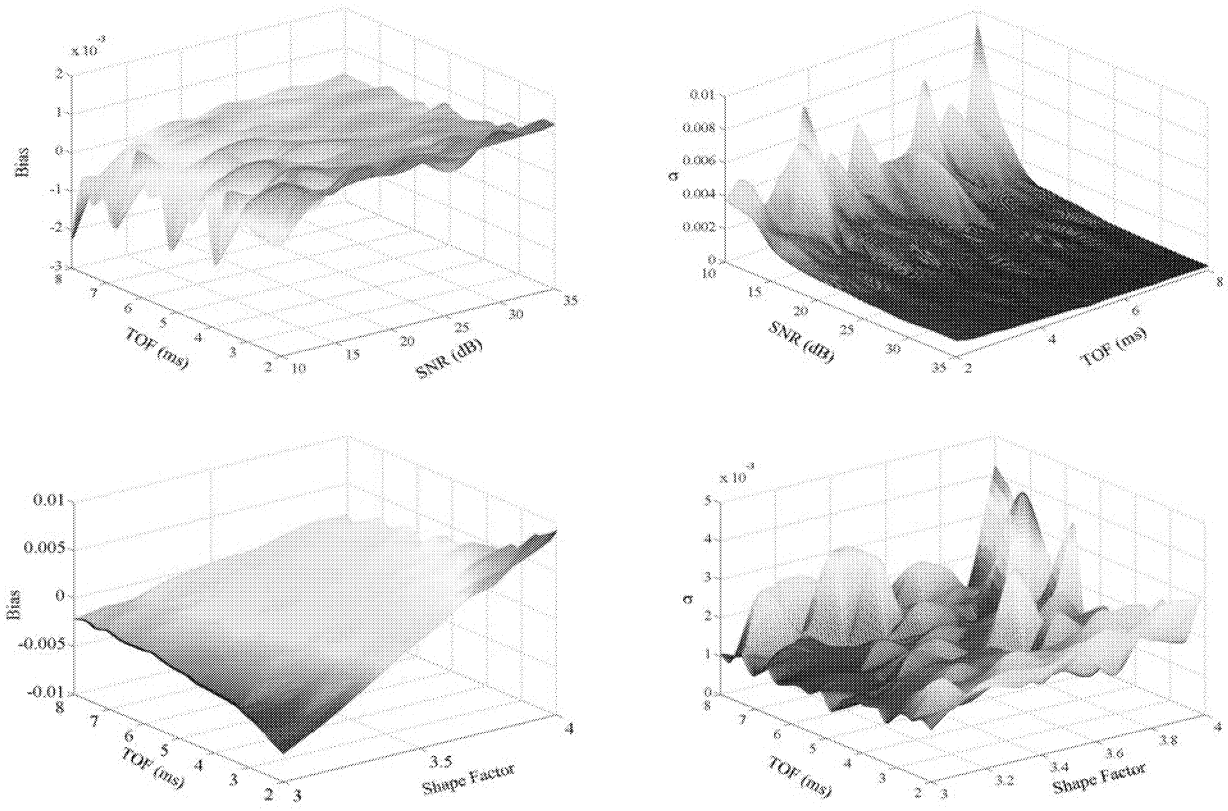


Fig. 7 – Values of bias and σ obtained in the numerical tests.

been evidenced; large variations of the shape factor do not seem to significantly degrade the quality of the results.

- Low-pass filtering in the I/Q demodulation stage allows the effect of noise to be suitably rejected.

CONCLUSIONS

A digital signal-processing algorithm for accurate ultrasonic TOF estimation has been presented. The performance of the digital signal-processing algorithm has been investigated with numerical tests. A statistical experiment design-based technique has been used in order to obtain significant results even though the test burden has considerably been reduced. Obtained results in terms of bias and experimental standard deviation, σ, are satisfying. Further tests carried out to study in depth the effect of two most significant inaccuracy sources (i.e. SNR and α) has proven that bias and σ are always lower than 1% in the whole investigated range.

REFERENCES

Bishop, M.J.D, "Digital filters: a practising engineer's perspective", Digital Filters: An Enabling Technology (Ref. No. 1998/252), IEE Colloquium on , 1998, pp. 3/1 -3/6

Cai C., Regtien P. L., "Accurate Digital Time-of-flight Measurement Using Self-Interference", IEEE Trans. Instr. And Meas., vol. 42, n.6, Dec. 1993

Eren H., "Effect of impedance mismatching and time delays in ultrasonic measurements," in Proc. IMTC'95, Waltham, MA, Apr. 24–26, 1995.

Oppenheim A. V. and Shafer R. W. "Discrete-Time Signal Processing", Englewood Cliffs, NJ: Prentice-Hall, 1989.

Montgomery D.C., "Design and analysis of experiments", John Wiley & sons, 1997

Sabatini, A.M., "Correlation receivers using Laguerre filter banks for modelling narrowband ultrasonic echoes and estimating their time-of-flights", IEEE Trans. on Ultras. Ferroel. and Frequency Control, vol. 44/6, Nov. 1997, pp. 1253 –1263.

Sabatini, A.M., "A digital signal-processing technique for compensating ultrasonic sensors", IEEE Trans. on Instr. and Meas., Vol.44/4 , Aug. 1995, pp. 869 -874

Sydenham P. H. and Thorn R., "Handbook of Measurement Science" New York: Wiley, 1992.

AUTHORS' BIOGRAPHIES

ROSARIO SCHIANO LO MORIELLO was born in Castellammare di Stabia, NA, Italy, on December the 21th, 1975. He received the M.S. degree (cum laude) in Materials Engineering from the University of Naples Federico II. He is, actually, attending the second year of PhD study course.

MODELLING AND SIMULATION OF A BIOLOGICAL PROCESS (NEURAL ACTION POTENTIAL) WITH HYBRID TOOLS USED IN COMPUTER SCIENCE

Jens Kohlmeyer[1], Stefan Sarstedt[2] and Wolfgang Mader[3]
Department of Programming Methodology and Compiler Construction and Department of Neurobiology
University of Ulm
89069 Ulm, Germany
[1]jkohlmey@bach.informatik.uni-ulm.de
[2]sarstedt@acm.org
[3]wolfgang.mader@biologie.uni-ulm.de

ABSTRACT

This paper shows an approach to model a complex biological process with hybrid tools used in computer science, in particular in software engineering.
The goal was to to capture the behavior of an axon with its neural action potentials by combining state based description techniques and continuous equations. The axon was modelled and the behavior simulated afterwards.
On the basis of tests which are also used in "real" biological experiments it was shown that the model is accurate enough to reproduce biological reality. The results of the model resemble the results of experiments in biology with high accuracy.
Our model and its simulation are used in a practical course at the University of Ulm which is aimed to help the students gain a better understanding of complex processes in biology.

INTRODUCTION

We made efforts to research the possibility of using software engineering tools to model complex biological structures and their behavior. The components of the axon needed to be modelled in detail, and the collaboration of these components during the stimulus conduction were simulated by a tool. We wanted the parameter settings to be easily manipulable to allow the user experimentation with the model. As mentioned before, the model is used in a practical course for education in which students work with the biological experiments and in parallel with the model on a computer.
The paper consists of two parts. In the first part the modelling of the components of an axon is described; this includes the tools that were used and the setup of the model as a whole. The second part deals with the simulation and the usage in the practical course. The results are illustrated by an example of an experiment.

MODELLING AN AXON

First we will give a short introduction of neurons, the axon and the action potential.
For the research done so far only the signal transfer within a singular neuron was examined. The neuron sends a signal called the *action potential* down the axon, away from the cell body. Neurons send signals electrochemically. The chemicals which are electrically charged ions cause an electric signal at the cell membrane by passing the membrane through ionic channels. For more information on these topics please see [Alberts, 2002], [Ude, 1994] and [Hille, 1992].
When modelling an axon two components at top level are very important. The first of these components are the nodes of Ranvier. Only at these small nodes electric activity takes place. The second component is the transfer of the voltage (spike propagation) from one node of Ranvier to another; this is called *stimulus conduction*. The electric impulse jumps from one node to the next. This is called *saltatory conduction*.

The first section in this chapter deals with the modelling of the nodes of Ranvier, the second section talks about the spike propagation on the axon.

Modelling a node of Ranvier

When we examined the procedure of an action potential it became evident that the use of hybrid models would be the best way to model an axon and its behavior. The combination of discrete events in statecharts and the continuous calculation of parameters is well suited for modelling this biological process. The itemized components of the axon are modelled with statecharts, and

the calculation of the results is made by a tool using continuous differential equations.
We chose to use Stateflow and Simulink from *The Mathworks, Inc.* Both tools are integrated into MATLAB and work together very well. Simulink offers a simple way of simulating a model.

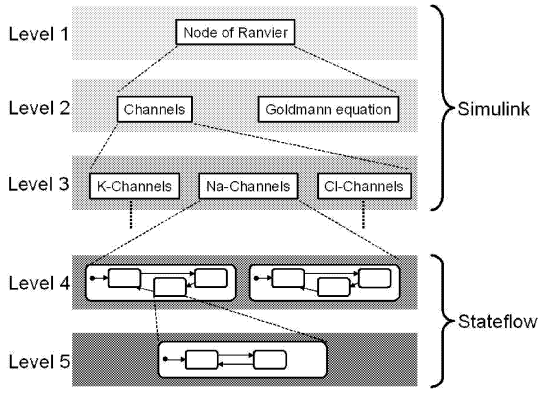


Figure 1: Abstract of the model.

The model of a node of Ranvier in our work is composed of five hierarchical levels, as shown in Figure 1. The three top levels are modelled in Simulink, and the two bottom levels which contain the singular components of an axon are modelled in Stateflow. The five levels are introduced from top to bottom in the following.
The topmost level contains a single node of Ranvier with an incoming impulse (for example a mechanical stimulus) and an outgoing calculated voltage value (see figure 2).

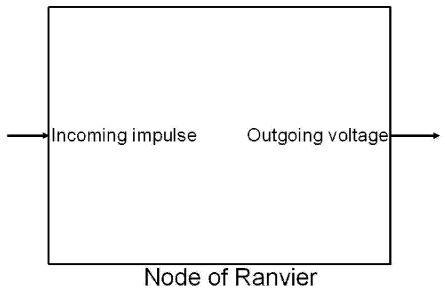


Figure 2: Level 1 – a node of Ranvier

It is simple to connect several nodes of Ranvier by structuring the model in this way. Information about the transfer of the voltage follows in section .
In the second level the voltage is calculated with the Goldmann equation. The potentials of nerves are mainly made up from interactions of three ions: Na^+, K^+ and Cl^-. The Goldmann equation uses the intracellular und extracellular concentration and the permeability of the ions.

The permeability of an ion is composed of the sum of the permeability of every single channel. It describes how many ions pass the membrane through the channels per time unit.

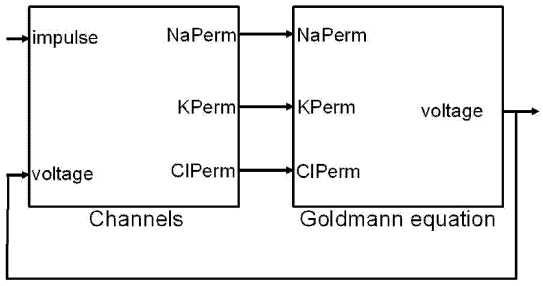


Figure 3: Level 2 – model of node of Ranvier

Figure 3 shows the Simulink model at level two. On the left there is a block called "Channels", on the right is the "Goldmann equation" block. In this block the calculation of the voltage takes place. The "Channels" block contains the channels of the axon, namely the sodium-, potassium- and the chlorine-channels. The stimulus from level one and the calculated voltage are directed to the block, as can be seen in the bottom part of the figure. The channels in an axon are controlled by the voltage, and therefore the voltage has to be redirected to the channels to control them. This leads to a problem: the channels depend on the very voltage which in turn depends on them. Simulink provides an elegant solution for this problem with the block called "unit delay". The calculated value is directed to the next block with a delay of one simulation step. The channels charge their permeability, the Goldmann equation uses these permeabilities to calculate the voltage, and one step later in the simulation this value is directed to the channels. In the very first step the voltage is presumed to be known.
The third level in Simulink contains the block for the different ionic channels (see figure 1). In this level the linking of Stateflow and Simulink is arranged; the Stateflow blocks are inserted in the Simulink model.

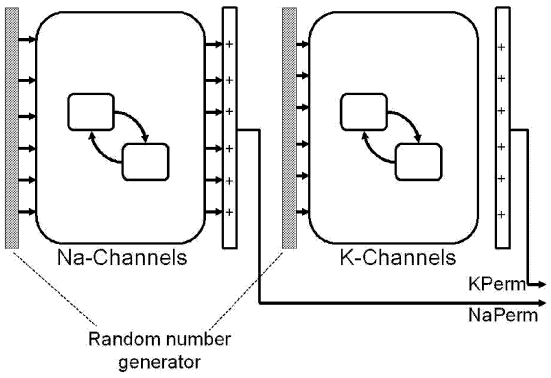


Figure 4: Level 3 – model of the block "Channels"

Figure 4 shows the model at level three. There are two kinds of exemplary channels in the model, the Na^+- and K^+-channels. The permeability of the channels modelled in the blocks is an outgoing parameter and is summed up for the permeability of the ion. The grey blocks are random number generators. They deliver a random number for every single channel to make it possible to model the random behavior of the channel. This means that the behavior of a channel is controlled by chance, just like in the biological reality. The channels show a flickering behavior [Alberts, 2002].
On the basis of the sodium channels the levels four and five are now exemplified. The other ionic channels have been modelled analogically, adapted to their specific behavior.
Figure 5 shows some modelled sodium channels in a Stateflow model. The channels are framed with a dotted line, which means that they execute in parallel. In a single channel the states are framed with a solid line. These states are mutually exclusive, i.e. the channel is exactly in one state at a certain time. A channel can either be "open", "closed" or "inactive" (further information in [Beck, 2001]). The transitions are conditioned by the behavior of the channel; for example, if the channel in the model is in the state "closed" and the voltage reaches a certain value, it switches to the state "open". The transition in this case is conditioned by the incoming voltage.

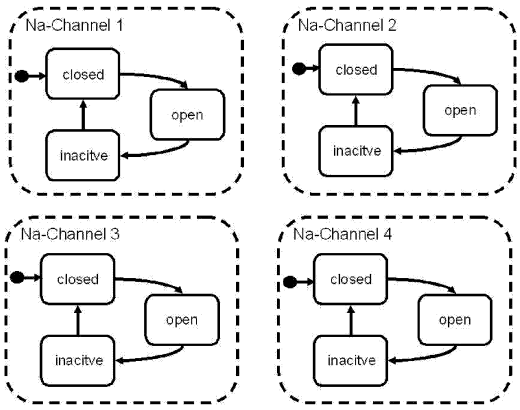


Figure 5: Level 4 – The sodium channels

The permeability of the channel depends on the state in which the model is at that moment (if the channel is "open", then the permeability is high, otherwise low). The states in level four of the model actually do not calculate the permeability of the channel, because the random behavior of the channel has not been considered. Therefore, these states are conceptual, they do not calculate anything.
The calculation of the permeability of the channel takes place in level five and depends on the conceptual state. Here, the random behavior is modelled. Figure 6 shows the "closed"-state of a sodium channel at level five.

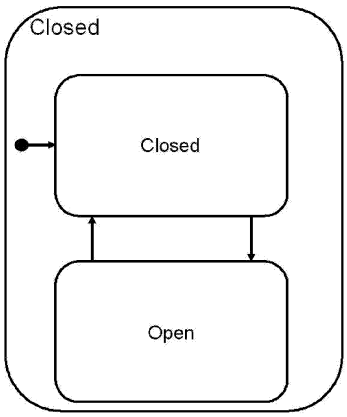


Figure 6: Level 5 – The "closed"-state of a sodium channel

In the states "open" and "close" the calculation of the permeability for the channel takes place. It is made up of the conductance of the channel and an additional factor. This factor is given by the states of the channels. The transitions are conditioned by the probabilities of the channels. In spite of the state "closed", the channel can open for a moment, just like in reality (the flickering behavior).

The spike propagation

After we have now shown the modelling of a node of Ranvier, we will now look at the transfer of the voltage from one node to another. The following formula (1) calculates the incoming voltage at a node of Ranvier.

$$E_m = E_0 * \exp -\frac{S}{\lambda} \qquad (1)$$

E_m is the incoming voltage at the next node. It is calculated from the outgoing voltage of the preceding node (E_0), the distance of the nodes (S) and the membrane length constant λ. For additional information about this formula see [Alberts, 2002] and [Kohlmeyer, 2003].
When using the membrane length constant, the diameter and the myelin sheath of the axon are included in the formula. Thus, all fundamental parameters for the spike propagation are covered: the distance of the nodes, the thickness of the axon and the thickness of the myelin sheath around the axon. These factors determine the transfer of the voltage and they are adjustable in the model.
The formula is modelled as a Simulink block and can easily be included in the model: between two nodes of Ranvier the block "transfer" is inserted. It is possible to simulate an axon with several nodes in this way.

THE SIMULATION AND THE USAGE IN A PRACTICAL COURSE

It was necessary to compare the simulated behavior of the model with the behavior observed in experiments to see how good the model works. Therefore, it should be able to copy biological experiments with the simulation and to compare the results from the simulation with realistic results obtained in experiments.
In order to make it useable in the practical course, we tried to make the simulation of the experimental environment as close to the reality as possible. In [Hitt, 2002] several experiments with neurons are characterized.
The voltage applied to the axon (in other words, the stimulus) can be modified in steps of 0.1 mV during the simulation. The level of the stimulus is therefore adjustable. The stimulus is not permanently conducted to the axon, but only for short moments in certain intervals. The duration of the stimulus and the intervals are tunable as well.
With Simulink the simulation is quite easy. Simulink creates the simulation on its own, and the values of the parameters can be observed with so-called monitors placed in proper locations. Particularly interesting for the model are the voltage and the permeability of the ions.
In the practical course at the University of Ulm the students basically make experiments with the distance of the nodes, the thickness of the axon and the thickness of the surrounding myelin sheath. They test how these parameters affect the behavior of the axon. Therefore, these parameters can be adjusted conveniently and individually.
Typically, at the beginning of a session the students in the practical course do some experiments with real dissected axons. Afterwards, they work with the simulation at the computer. Through this combination of biological experiments and the usage of the model the students are able to watch real action potentials on one hand, and on the other hand they have the possibility to adjust those parameters in the simulation that can not be changed in the experiments, for example the axon diameter or the thickness of the myelin sheath.

In the following, we will describe an example of an experiment with the simulation. It is tested how the distance between the nodes of Ranvier affects the spike propagation along the axon. In the model are two nodes; the distance between them is variable. For this experiment the diameter of the axon is 10 μm and the thickness of the myelin sheath is 1 μm.
Figure 7 shows the results of two tests. The voltage curve at both nodes of Ranvier is displayed on the monitor. In test A (figure 7 A) the distance between the nodes is 1 mm, in test B (figure 7 B) 3 mm. The pink curve shows the course of the voltage for the first node, and the yellow curve for the second node during an action potential (see [Helmich, 1998] for an action potential on an oscilloscope). The curves in A are closely together, whereas in B there is more time (in other words simulation steps) between the curves. The explanation is simple: in test B the incoming voltage is much less then in test A, because the distance is bigger. The more distance there is between the nodes, the less voltage will reach the second node. Therefore, the second node reacts more slowly in test B as in test A, and the elapsed time between the action potentials of the nodes is longer.
By modifying the distance of the nodes the students gather information about how this parameter affects the behavior of the axon. As a result of the multifaceted options of settings in the model a lot of experiments can be duplicated:

- diversification of the axon diameter
- diversification of the thickness of the myelin sheath
- the effect of poison on the axon (for example, blockading the sodium channels)
- determination of the relative and absolute refractory period
- the influence of the probabilities of the channels on the behavior of the axon

We carried out these experiments with the model during the simulation and compared the results with known biological results. It turned out that the model matched reality in all aspects in a sufficient way [Kohlmeyer, 2003]. The students have a simple and fast tool to examine the influence of different parameters on the behavior of the axon. It is not possible to change these parameters in biological experiments, so that the use of this model provides further opportunities to analyze the behavior.
There are some difficulties when modelling a biological process: at first, all the basics of the process have to be known. The process of a neural action potential is well known so that data about it is readily available. However, the number of modelled channels presents a problem. Not as many channels as there are in real axons could be modelled due to restricted computing power. The more channels are modelled, the slower the simulation runs. The number of channels that we used in our model however, was still sufficient to achieve good results. We modelled 460 channels per node of Ranvier: 400 Na^+-, 40 K^+- and 20 Cl^- channels, which corresponds to the actual ratio of channels in a real node of Ranvier.
The use of hybrid tools with graphic programming

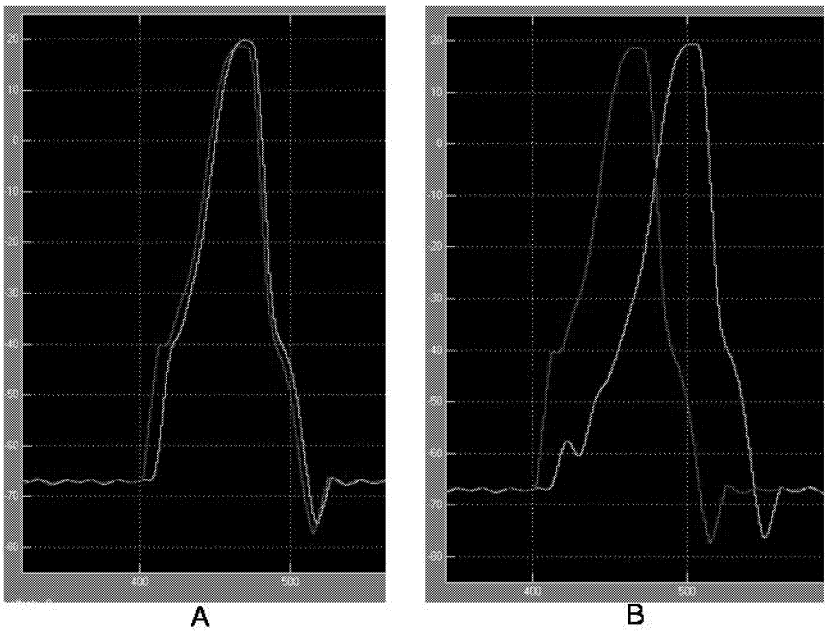


Figure 7: Comparison of the voltage depending on the distance of the nodes of Ranvier

has some benefits for the users, too. Non-computer-scientists seem to understand this modelling technique quite easily and fast. The understanding of statecharts is facilitated.
In Stateflow the statecharts can be monitored during the simulation. The fact that the user can monitor how single components act during an action potential and that the chronological coherence of the components can be seen highly benefits the understanding of the whole process. For the learning and understanding of a complex biological process the traceable results of the simulation are a great advantage.

CONCLUSION

Hybrid tools such as the combination of Simulink and Stateflow are well suited to model and simulate biological processes like a neural action potential. In spite of little difficulties, good results in modelling of biological components and their interaction are attainable.
The use of this modelling technique in education and science offers great advantages for students and research personnel.
The model and its simulation are used successfully in a practical course at the University of Ulm.

References

[Alberts, 2002] Alberts, B. (2002). *Molecular biology of the cell.* Garland Science.

[Beck, 2001] Beck, E.-G. (1997-2001). Biologiekurs 2001 Klasse 12.
http://www.merian.fr.bw.schule.de/Beck/skripten/12/bs12.htm.

[Helmich, 1998] Helmich, U. (1998). Neurobiologie.
http://www.drd.de/helmich/bio/neu/home.htm.

[Hille, 1992] Hille, B. (1992). *Ionic channels of excitable membrans.* Sinaver Associates.

[Hitt, 2002] Hitt (2002). Versuchsprotokoll – Tierphysiologischer Kurs für Studierende der Bioinformatik.
http://www.jainek.de/Protokoll%20Nerv.pdf.

[Kohlmeyer, 2003] Kohlmeyer, J. (2003). Modellierung und Simulation eines biologischen Prozesses mit hybriden Beschreibungstechniken aus der Informatik. Universität Ulm.

[Ude, 1994] Ude, Joachim Koch, M. (1994). *Die Zelle. Atlas der Ultrastruktur.* Gustav Fischer Verlag.

METVIS: A TOOL FOR DESIGNING AND ANIMATING METABOLIC NETWORKS

E. Qeli, B. Freisleben
Dept of Math.& Comp. Science, University of Marburg
Hans-Meerwein-Str., D-35032 Marburg, Germany;
{ermir,freisleb}@informatik.uni-marburg.de

D. Degenring, A. Wahl
Institute of Biotechnology 2, Research Center Jülich,
D-52425 Jülich, Germany;
{d.degenring,a.wahl}@fz-juelich.de

W. Wiechert
Institute of Systems Engineering, University of Siegen,
Paul-Bonatz-Str. 8-11, D-57068 Siegen, Germany;
wiechert@simtec.mb.uni-siegen.de

ABSTRACT

Metabolic networks are an important research area in the systems biology field. Several tools have been built to design and simulate metabolic network models. These tools range from simple command line simulators to elaborate graphical systems which not only allow to design and simulate metabolic networks, but also to visualize the simulation results. Typically, the visualizations are static snapshots of the data generated during a simulation. In this paper we present MetVis, a new tool for designing and animating metabolic networks. In contrast to previous approaches, MetVis allows to graphically animate the dynamic evolution of a modeled cell's metabolic properties based on produced simulation data. Relevant design and usage issues of MetVis are discussed, and its implementation using Java and XML is described.

KEYWORDS

Visualization, Animation, Metabolic Engineering, Metabolic Networks, Systems Biology, Java, XML

INTRODUCTION

The aim of metabolic engineering is the modification of the metabolism of biological cells towards specific goals by genetic manipulations (Bailey 1991, Stephanopoulos and Sinskey 1993). A key technique for understanding the behaviour of an observed cell is the modelling aproach. Usually, models are built to describe the stationary and dynamic behaviour of the metabolic network (i.e. the biochemical reaction network) of the central metabolism of a cell. Such models are based on measured and published data, and simulations are usually carried out using deterministic models gained from balancing the individual metabolic pools and connecting them by the kinetic expression of the enzymes. However, simulation alone is not sufficient to gain an understanding of metabolic networks. Appropriate techniques must be provided to analyze and interpret the possibly large volume of data generated by the simulations. Data visualization is a common technique that is offered by almost all currently used software tools supporting metabolic engineering. Typically, the data produced by the simulations are visualized using 2D plots to show snapshots of the trajectories of the different metabolite pool concentrations and reaction flows.

In this paper we present MetVis, a tool which not only allows to design and statically visualize snapshots of the behaviour of metabolic network models, but also allows to graphically animate the dynamic evolution of a modeled cell's metabolic properties based on produced simulation data. Thus, the novelty of MetVis compared to alternative approaches is to employ computer animation methods to improve the understanding of metabolic network models. The paper discusses relevant design and usage issues of MetVis, and describes its implementation using Java and XML.

The paper is organized as follows. The second section presents work on related tools for designing and visualizing metabolic networks. The third section describes the functionality and usage of MetVis in detail. The fourth section briefly discusses some relevant implementation issues and the fifth section shows experimental results obtained by MetVis. The sixth section concludes the paper and outlines areas for future research.

RELATED WORK

Several tools have been developed to make the design and analysis of metabolic networks easier. However, not all of them allow the graphical design and visualization of metabolic network models. In the following, we will give a short overview of most commonly used tools in this area, which offer the possibility to graphically design metabolic network models.

JDesigner

A well known tool for designing metabolic networks is JDesigner (Hucka et al. 2001a). The software is written in the programming language Delphi and is an open source project distributed under the LGPL license. JDesigner is part of an extensible software framework which also includes a simulator called Jarnac. It allows to import and export files for-

matted according to an extension of the Systems Biology Markup Language (SBML) (Hucka et al. 2001b) which is a common representation language for storing biochemical models. SBML is based on XML and contains structures for representing compartments, species and reactions, as well as optional unit definitions, parameters and rules (i.e. constraints). A separate module of JDesigner has the responsibility to visualize snapshots of the trajectories of species concentrations and reaction flows.

Gepasi

GEPASI (Mendes 1993, Mendes 1997) is a MS Windows software tool for modelling chemical and biochemical reaction networks. It allows to produce trajectories of the metabolite concentrations and obtain a steady state (if it does exist). Import/export from/into SBML is also supported. GEPASI has a graphical user interface to input metabolic network information, but does not allow the interactive design of metabolic network models. The trajectories of metabolite concentrations are visualized using Gnuplot.

Bioq

BIOQ (Bioq) is another tool for modeling chemical and biochemical reactions networks. It includes a GUI interface to design the network and a simulation engine to simulate the behaviour of the system. BIOQ's features for designing a model are limited, and the results of a simulation can be viewed in a simple plotting window. BIOQ uses a proprietary format to store models and does not have an interface to import the SBML format. BIOQ is available for free both for Unix and MS Windows.

FluxAnalyzer

The FluxAnalyzer (Klamt et al. 2003) is a package for the commercial MATLAB system available from Mathworks, Inc. The tool is multi-functional in the sense that it allows e.g. metabolic flux analysis, structural and pathway analysis, and network composition. To visualize metabolic networks, so called interactive flux maps are used, which represent the results of a simulation of a metabolic network in an interactive manner. These interactive flux maps are represented by text boxes with attached fluxes which have different colors for defined rates, calculated rates, and non-calculatable rates. However we have to mention that in this case we have to deal with stationary metabolic flux distributions.

Virtual Cell

The Virtual Cell (Lowe and Schaff 2001) is an elaborate software framework implemented in Java, intended to function also in the WWW. It has a design module which allows the interactive design of network models. The Virtual Cell stores the data in a native format (VCML), but also allows export to SBML or CELLML, another language for cell modelling. The results of a simulation can be viewed in a visualization window in form of a trajectory or exported to text format.

Further Approaches

There are several other tools for metabolic network design and visualization not described in more detail here (Mauch et al. 2001, Adachi et al. 2001, Dickerson et al. 2001, Vass et al.). Some of them are commercial tools (Mauch et al. 2001) and therefore it is difficult to find technical information about them without buying and installing them. Furthermore, there are other non-commercial tools for which no published information exists or the information published is not sufficient to understand the relevant functionalities of the tools.

THE METVIS APPROACH

Most of tools described above consist of the following modules:

1. Network design module
2. Simulation module
3. Visualization module

MetVis is also part of a framework which consists of these three modules. MetVis provides an easy to use, comfortable graphical user interface for graphically designing metabolic networks, the Metabolic Modeling Tool (MMT) described in (Haunschild et al. 2002, Hurlebaus 2002) is our simulation module, and MetVis uses the output produced by MMT to visualize and animate the results of a simulation beyond the typical static representations of the trajectories of metabolite concentrations. In the following, the complete process from the design of a metabolic network model to the visualization of the results of the simulation will be presented.

Design of Metabolic Networks Models

The design of a metabolic network model in MetVis is similar to other tools that offer this feature (Hucka et al. 2001a, Lowe and Schaff 2001). The desired metabolic network component (i.e. specie, reaction node, reaction flow, compartment and inhibitor/activator) to be drawn is simply chosen from the toolbar offered by MetVis (Fig. 1). A simple mouseclick is required to draw it on the screen in case of a reaction node or a specie, whereas for a reaction flow the source species, the reaction node and the destination species have to be clicked in the corresponding sequence. For reactions and species, additional information should be provided. For example, the name is the most important input as it creates the logical connection with the output of a simulation. This information can be specified explicitly or taken from an already existing model. Figure 1 shows the design of a simple reaction with two educts and three products. The species are represented by a square, and reactions are represented by a rhombus. Reaction and specie properties can be defined in a special dialog in the case when the model file will be

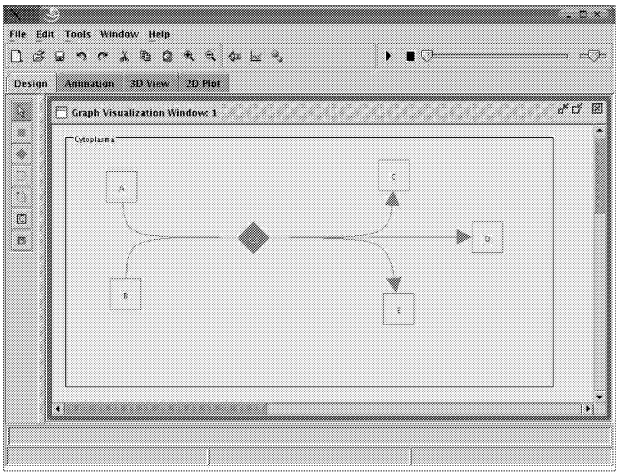


Figure 1: A Simple Reaction with Two Educts and Three Products

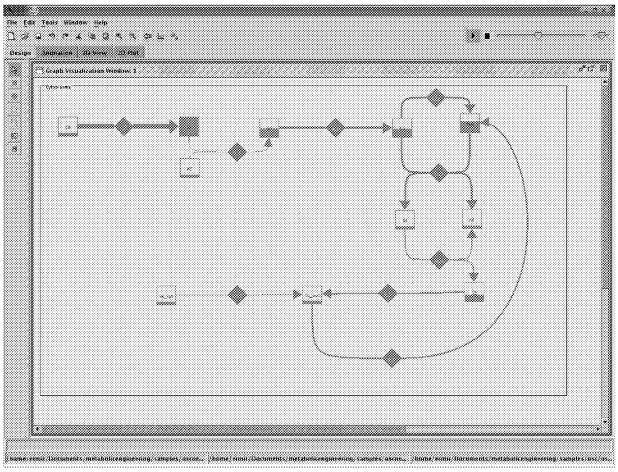


Figure 2: A Screenshot in the Course of an Animation

exported or read from an existing model file. After finishing the design, the work can be saved, resulting in two XML files, one containing the model file and the other containing its graphical representation.

Simulating Metabolic Networks with MMT

The architecture and functionality of our simulator MMT has been described in (Haunschild et al. 2002, Hurlebaus 2002) and thus is beyond the scope of this paper. However, some features important to the visualization/animation offered by MetVis should be mentioned. MMT is a command line program for Linux which allows the distributed simulation of metabolic networks with model variants on a network of workstations. It consists of two parts: a simulator generator and the actual proper simulator. The simulator generator takes as its input an XML file representing a metabolic network model and creates the simulator for this model. The format of the model file is a dialect of SBML, as MMT has features which go beyond the possibilities of SBML, such as model variants that allow simultaneous simulation for a set of similar models. The generated simulator which is tailored to the model file can then be instructed to generate the data needed for the subsequent animation. The output from the simulator can be customized to obtained different formats such as XML, CSV, etc.

Animation of Metabolic Networks

To run an animation, the results of a simulation must be delivered in form of a CSV (Character Separated Values) file. The information stored there contains the concentrations of metabolites and flows of reactions varying over time. MetVis reads this information and stores it in an appropriate data structure for later use. The dynamic metabolic behaviour is expressed by an animation showing changing metabolite pool sizes and changing fluxes being represented by differently filled boxes and varying arrow widths, respectively, according to data generated by a simulation. In figure 2, we see the state of an animation of a sample metabolic network model after a certain time has elapsed. It is evident that boxes representing different metabolites have different fill volumes, showing the size of the metabolite at a specific time. The same idea applies to the reactions with the width of the Bezier curves representing reactions changing over time. To control the animation, two buttons (play/stop) and two sliders are provided. One of the sliders controls the time and the other one controls the speed the animation will run with.

Implementation of MetVis

MetVis consists of about 60 Java classes grouped in different packages. Figure 3 shows a simplified UML class diagram of MetVis, in which for space reasons only the most important classes are present. MainFrame is the main class

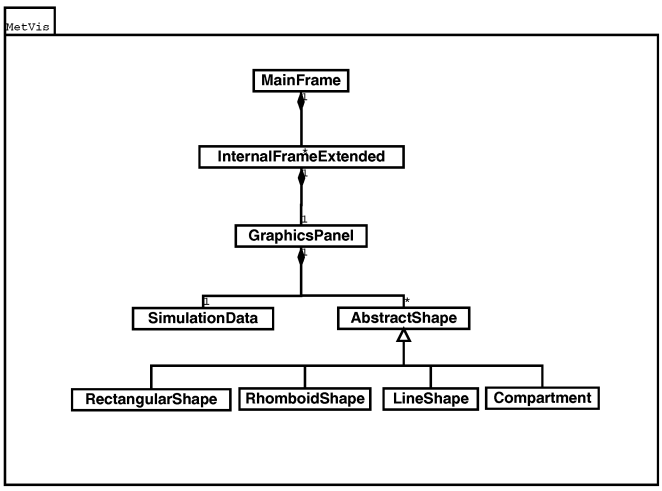


Figure 3: A Simplified UML Diagram of MetVis

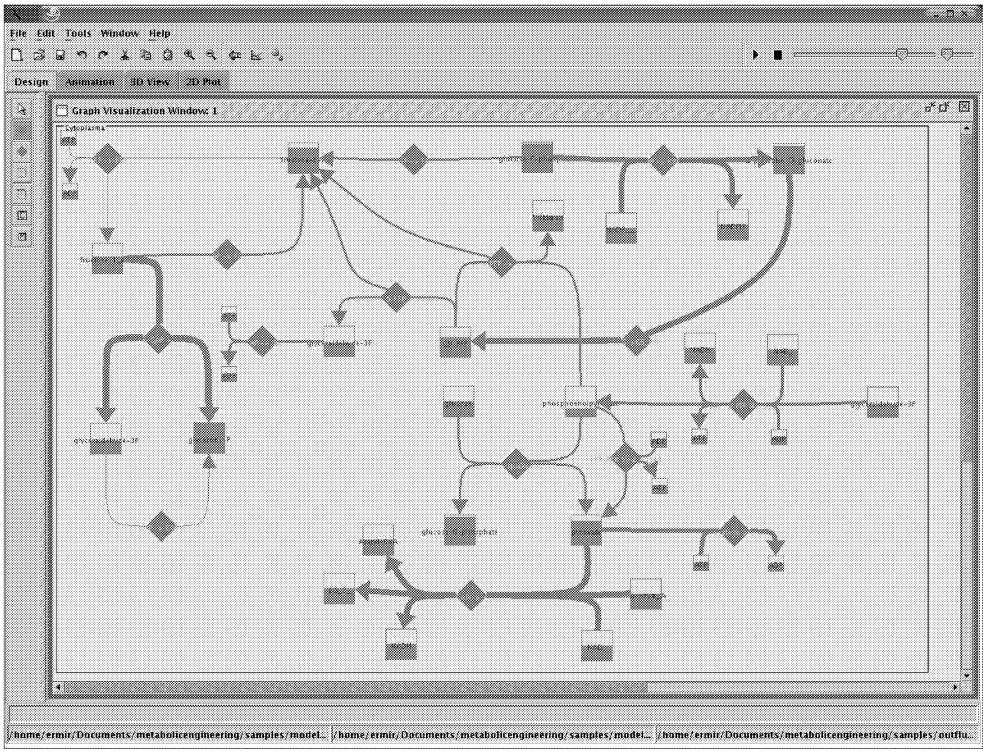


Figure 4: 531th Step of an Animation of Part of an E.coli Model

of the application. It can contain zero, one or more (MetVis is a multiple document interface (MDI) application) InternalFrameExtended objects, which represent windows where model design can take place. GraphicsPanel is the low level container of drawn objects and the simulation results. Drawn objects are represented by the base class AbstractShape, from which several classes are derived. The design of MetVis is based on modern software engineering methods such as design patterns (Gamma et al. 1994). In our case, the following patterns were used:

- Observer/Observable to allow notification between different objects
- Factory pattern to allow the creation of objects derived from the same class
- Model/View/Controller to support multiple windows
- Command to allow undo/redo

The SAX library is used to read XML model files, because it allows a more efficient processing of the file. The output of XML files is done as simple text to achieve a better performance with large files. A separate thread is employed to control the progress of the animation, in order to ensure good response times and high interactivity while playing/stopping/pausing the animation.

EXPERIMENTAL RESULTS

Currently, MetVis is being used by our project partners of the Institute of Biotechnology in the Research Center Jülich, Germany. Several models have already been designed. Among them is a model of part of the metabolism of E.coli, of which two randomly selected animation steps (the 531th step and the 670th step) are shown in figures 4 and 5, respectively. In figure 5, some of the differences to figure 4 are illustrated by the highlighted numbered rectangles. It is evident that the quantity of the metabolites Glyceraldehyde-3P (rectangle 2) and C5-PPP (rectangle 4) and reaction FBP -> GAP + DHAP(rectangle 1) are much smaller in the second screenshot in comparison to the first screenshot. The contrary happens with the reaction GAP->PEP BlackBox (rectangle 6), since its quantity is larger in the second screenshot. The analysis of the network is made much easier in this way. In addition to its animation features, MetVis offers the possibility to view the trajectories of concentrations of metabolite pools and reaction flows, as shown in figure 6. This is the type of visualization usually provided by the alternative tools discussed in the related work section.

CONCLUSIONS

In this paper, we have presented MetVis, a tool which supports the design, visualization and animation of metabolic networks. In contrast to previous approaches, MetVis allows to graphically animate the dynamic evolution of a modeled cell's metabolic properties based on produced simulation data. The functionality and implementation of MetVis was described, and experimental results showing the benefits of our approach were presented.

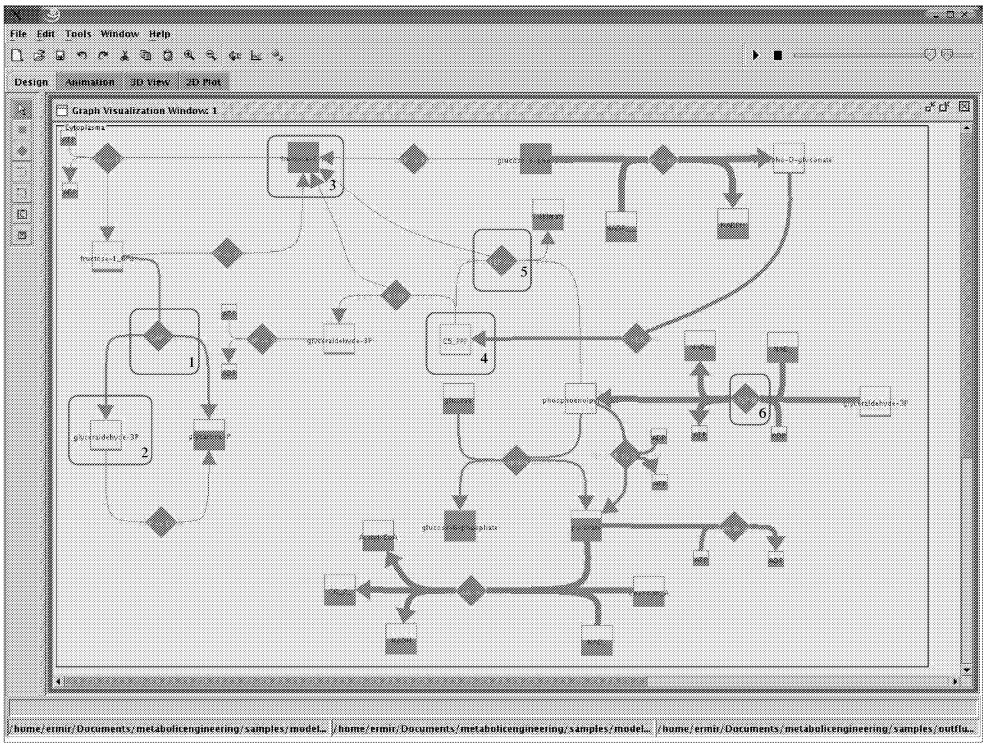


Figure 5: 670th Step of the Animation of Part of an E.coli Model with Some Outlined Differences over Previous Screenshot

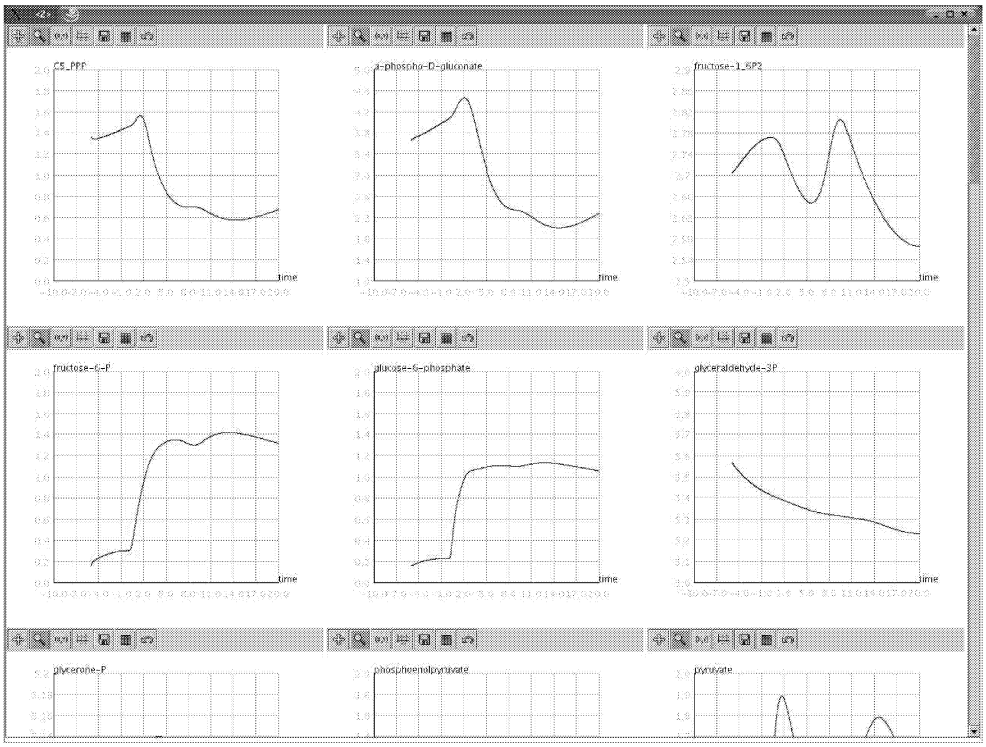


Figure 6: Trajectories of Metabolite Concentrations and Reaction Flows

There are several areas for future research. For example, it would be interesting to explore the possibility whether 3D graphical animations (instead of the 2D views provided currently) would add to the understanding of metabolic network

models. Another direction we are working on is to enrich the visualization abilities of our tool by visualizing the sensitivity matrices resulting from studying how the variation in the output of a model can be attributed to different sources of variation in its input. Since their dimensions in metabolic engineering can be relatively large, e.g. with a typical dimension of 20 x 10 and also calculating the time variation we have 20 x 10 x 100 elements to visualize, our future work in this direction consists of developing a tool to analyse and visualize these sensitivity matrices in a time-independent manner.

ACKNOWLEDGEMENTS

This work is financially supported by Deutsche Forschungsgemeinschaft (DFG, Schwerpunktprogramm 1063) and by Deutscher Akademischer Austausch Dienst (DAAD, Stability Pact for South Eastern Europe).

REFERENCES

Adachi, T.; N. Matoba and H. Kurata. 2001. "BIOCAD project for constructing a biological system in silico", *Journal of Genome Informatics* 12, pp.284-285.

Bailey, J.E. 1991. "Toward a science of metabolic engineering", *Science*, volume 252, pp. 1668-75.

Bioq: http://leonardo.ls.huji.ac.il/~parnas/Bioq/bioq.html

Dickerson, J.A; D. Berleant; Z. Cox; W. Qi and E. Wurtele. 2001. "Creating metabolic network models using text mining and expert knowledge" *In the Proceedings of Atlantic Symposium on Molecular Biology and Genome Information Systems and Technology(CBGIST 2001)*, pp. 26-30, Durham, North Carolina.

Gamma, E.; R. Helm; R. Johnson and J. Vlissides. 1994. *Design Patterns: Elements of reusable object-oriented software*, ISBN 0201633612, Addison Wesley.

Haunschild, M.D.; B. Freisleben; W. Wiechert and R. Takors. 2002. "Distributed simulation of metabolic networks with model variants" *In Proceedings of 16th European Simulation Multiconference*, pp. 436-440, SCS Press, Darmstadt, Germany.

Hucka, M.; A. Finney; H. Sauro and H. Bolouri. 2001. "The ERATO Systems Biology Workbench: An Integrated Environment for Multiscale and Multitheoretic Simulations in Systems Biology", *Foundations in Systems Biology*, MIT Press. http://www.cds.caltech.edu/~hsauro/JDesigner.htm.

Hucka, M.; A. Finney; H. Sauro and H. Bolouri. 2001. "Systems biology markup language level 1: Structures and facilities for basic model definitions", Technical Report MC 107-81, California Insitute of Technology, Pasadena, CA 91125, USA.

Hurlebaus, J. 2002. "MMT-A pathway modeling tool applied to data from rapid sampling experiments", *InSilico Biology*,Vol. 2, 0042.

Klamt, S.; J. Stelling; M. Ginkel and E.D. Gilles. 2003. "FluxAnalyzer: exploring structure, pathways, and flux distributions in metabolic networks on interactive flux maps", *Bioinformatics* 19(2), pp. 261-269.

Loew, L.M. and J.C. Schaff. 2001. "The Virtual Cell: A software environment for computational cell biology", *Trends in Biotechnology* 19, pp. 401-406.

Mauch, K.; S. Buziol; J. Schmid and M. Reuss. 2001. "Computer aided design of metabolic networks", *American Insitute of Chemical Engineers(AIChE) Symposium Series*, Tucson, Arizona.

Mendes, P. 1993. "GEPASI: A software package for modelling the dynamics, steady states and control of biochemical and other systems", *Comput.Applic.Biosci.9*, pp.563-571.

Mendes, P. 1997. "Biochemistry by numbers: simulation of biochemical pathways with Gepasi 3", *Trends Biochem.Sci.*, pp.361-363.

Stephanopoulos, G. and A.J. Sinskey. 1993. "Metabolic engineering–methodologies and future prospects",*Trends in Biotechnology*, volume 11, pp. 392-6.

Vass, M.; C.A. Shaffer; J.J. Tyson; N. Ramakrishnan and L.T. Watson. "The JigCell model builder: A tool for modeling Intra-Cellular Regulatory Networks", *Bioinformatics*, In Press.

SIMULATION OF DISEASE CONTAGION

MODELLING THE EFFECT OF INFORMATION FEEDBACK ON THE SPREAD OF DISEASE: A CASE STUDY ON THE EBOLA VIRUS

Dr. Bernadette O'Regan & Prof. Richard Moles
Department of Chemical and Environmental Sciences
University of Limerick
Ireland
E-mail: bernadette.oregan@ul.ie

KEYWORDS

Ebola, Computer Simulation, Modelling, System Dynamics, Feedback, Information Delays

ABSTRACT

This paper describes the application of the tools and techniques of the system dynamics method to the problem of understanding the spread of disease using the Ebola virus as a case study. The main deliverable of this research is a computer simulation model in the system dynamics tradition. The essence of system dynamics is to act as a framework for formalising mental models of a problem. In this respect, the system dynamics simulation model presented here is a theory describing the structure of, and interrelationships between, the factors which impact an outbreak of the Ebola virus and the attempts to contain it.

The model, comprising 57 interrelated variables, is structured to represent a group of rural villages served by one local hospital, remote from regional and national medical laboratories. Such a structure typifies the circumstances of initial Ebola outbreaks in central Africa. Model output examines the probable impacts of changes in the system delays. These delays consist mainly of incubation delays, delays to disease recognition, delays in travelling to hospital, delays to inform higher health authorities and delays to involve the Centre for Disease Control in the US.

INTRODUCTION

The Ebola virus takes its name from the Ebola River in northern Congo, where it first emerged in 1976. In this outbreak, 318 people were infected and 280 died (Le Guenno and Galabru, 1997). On average the disease has an 80% fatality rate (Michie, 1999, Peters, 1997). Recent studies (Le Guenno and Galabru, 1997) have identified four strains of the virus[i], suggested by differences in mortality and clinical expression.

The contamination risk is related to contact with the body fluids of an infected person. This is most likely during the nursing of patients and the preparation of corpses for burial. Also, as the disease kills so rapidly the risk of an epidemic in developed countries is believed to be low (Sinha and Powell, 1996).

Outbreaks tend to spread from one isolated case and so individual behaviour and local customs play an important role in the progress of each outbreak of Ebola or a similar epidemic (Le Guenno and Galabru, 1997). Initial cases leading to secondary transmission to the person taking care of the infected person are called 'familial outbreaks'. The tendency so far has been for the spread to 'amplify' when infected people enter local hospital care without proper protective equipment or hygiene rules and then 'explode' as the virus spreads to general hospitals, as was the case in the Kitwit (Zaire) outbreak in 1995 (Shears, 2000, Peters, 1997). These hospitals tend to be ideal breeding grounds for disease due to the poor training of staff, inadequate staff levels, poor standards of hygiene, lack of analytical laboratory equipment and only basic medicine (Shears, 2000).

AVAILABILITY OF DATA

Recent outbreaks have confirmed data already acquired by the World Health Organisation (WHO) in 1996. Transmission is not airborne. Transmission requires close contact to a seriously ill patient which, in most situations, results in a low infectiousness rate. Therefore, isolation and a change in cultural customs can help in the containment of the virus. The incubation period varies from 3 days to 3 weeks and when the patient has a lethal form of the disease, death usually takes place between 6 and 10 days from the initial onset of the illness (Le Guenno and Galabru, 1997).

The best currently available diagnosis techniques, antigen capture on serum samples, require at least 3 days under the best conditions. This test can only be carried out at biosafety level 4 laboratories.

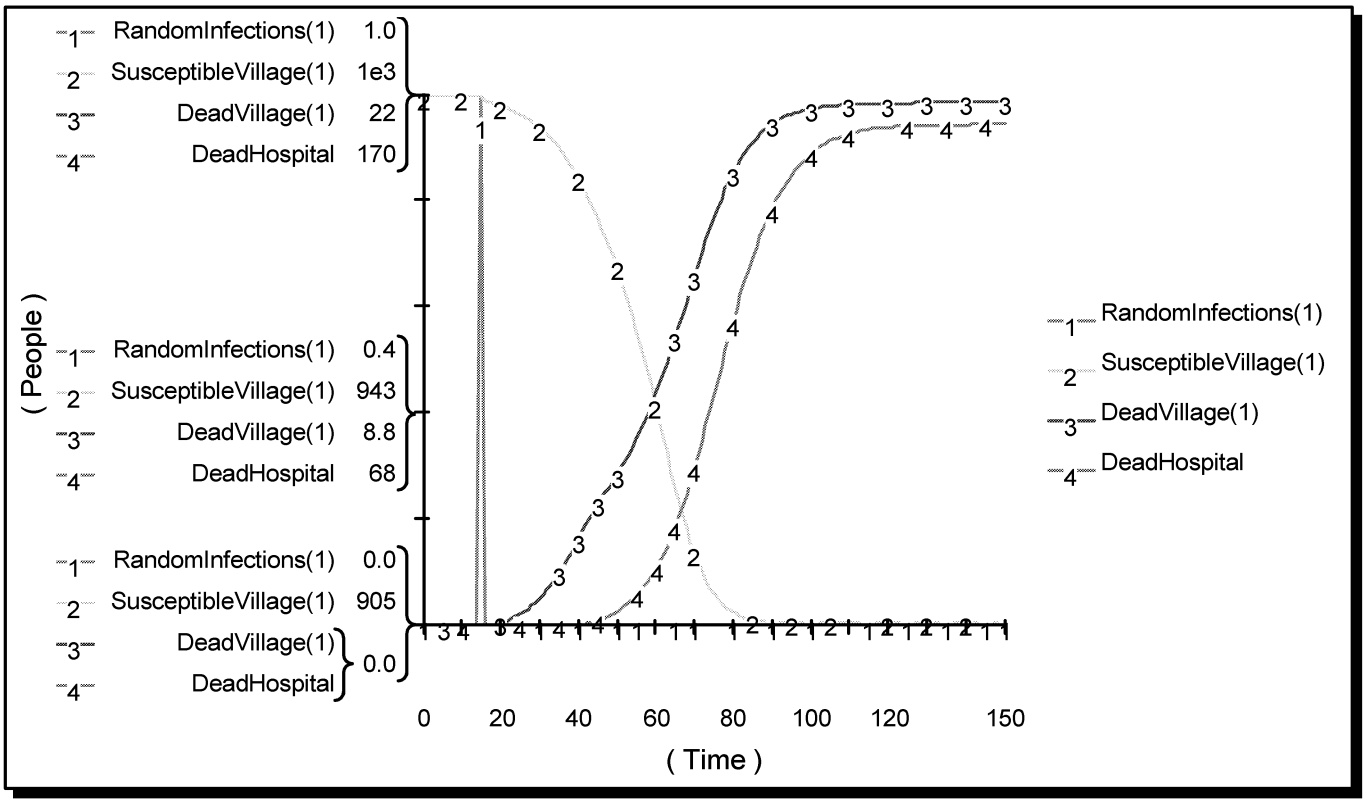


Figure 1: Output from the standard run of the model

Little if anything is yet known about the host of the virus, except for the curious fact that a number of outbreaks have occurred around October or November, suggesting a link to seasonal or annual cycles in the forests of central Africa. The most recent outbreaks are consistent with this pattern. The WHO is presently coordinating a project, which facilitates the observation and capture of species in chimpanzee territories of the Tai forest, where the chimpanzee population has already been hit by two separate outbreaks of the virus. The remainder of this paper describes a simulation model designed to capture the important dynamics of an Ebola outbreak. The model is developed in the system dynamics tradition, with emphasis on the importance of feedback and information delays.

MODELLING AN OUTBREAK

For the purposes of the model described here, it is assumed that the outbreak originates from a single isolated incident on day 15 of the 150 day simulation period. From Figure 1[ii] it can be seen that this random infection resulted in a total death toll of 192 people over the 150 days.

Figure 2iii shows the behaviour over time graphs for the spread of the virus in the local village. Note that the number of sick people declines briefly around day 40. This corresponds to the time when the sick start to move into hospital.

The model assumes the size of the susceptible population to be 1000 people. This is assumed to correspond to the size of a rural village. The rate at which other people become infected in the village is a function of how easy it is to become infected (the infectiousness rate), the rate at which an infected person comes in contact with other people (the contact rate) and the percentage of incubating and sick people already in the village, defined as follows;

RandomInfections + (ContactRate InVillage *
InfectiousnessInVillage * SusceptibleInVillage *
(Incubating InVillage + SickInVillage)
/ Total Population) (1)

It is assumed here that both incubating and sick people can pass on the virus. Furthermore, it is assumed that a susceptible person is equally likely to get infected from either an incubating or sick person. In practice, the virus is more likely to be transmitted through the nursing of very sick people but it could be argued that the contact rate would probably be higher for incubating people. In any case, these differences are abstracted away in this model but the model may be further enhanced if it was ever necessary to disaggregate these relationships.

ContactRateInVillage and InfectiousnessInVillage are more difficult to quantify, however, these variables are important so they must be included in the model. It is also necessary to estimate how many new people might an infected person living in a remote African village meet each day. The standard run of the model as shown in Figure 1 assumes a contact rate of 1, i.e. an infected person meets one new[iv] person each day. However, evidence suggests that once the cause of the epidemic is identified exposed populations are warned of the dangers of coming in contact with infected persons and so the contact is not a constant but instead declines over time and is defined as follows;

*ContactRateInVillage = 1 * (1 - BarrierNursing)* (2)

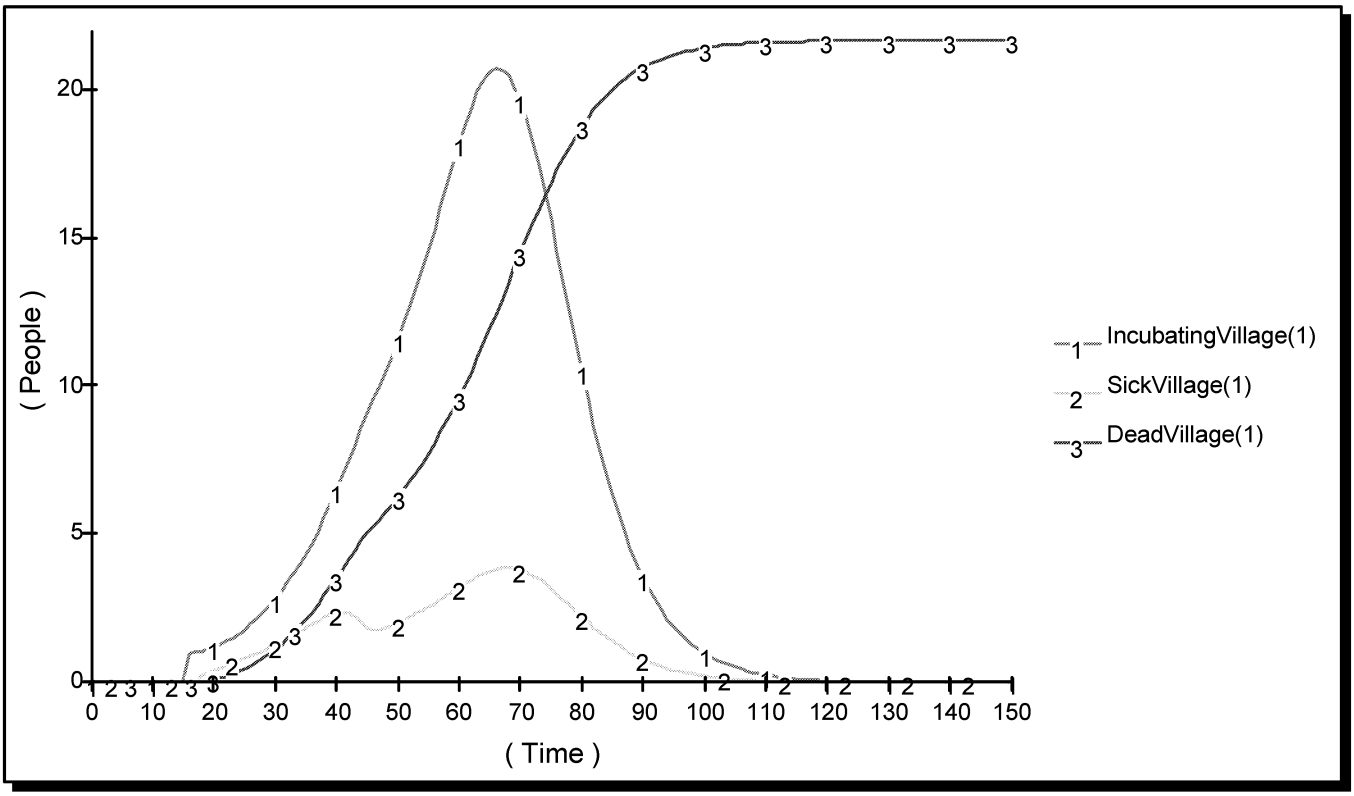


Figure 2: The spread of the virus in the local village

where *BarrierNursing*, expressed as a percentage, is taken to represent a type of disease containment using isolation techniques implemented by the national disease surveillance unit or the CDC during each Ebola outbreak to date.

The I*nfectiousnessRate*, meaning the infectiousness of each contact between a susceptible and an incubating or sick (infectious) person, is defined as 0.15. This figure was arrived at through extensive sensitivity analysis, based on available data from historical Ebola outbreaks.

A value of 8 days was chosen for the incubation period and the number of people getting sick each day in the village is then defined as;

IncubatingInVillage / IncubationPeriod (3)

Similarly, *DeathsInVillage* is defined;

*SickInVillage * (DeathFraction / LengthOfSickness)*(4)

where *DeathFraction* is assumed to be 80% and *LengthOfSickness* to be 6 days.

People who don't die, but remain in the village, become 'survivors', as defined by the *SurvivalFraction* (*1 – DeathFraction*). It is assumed for the purposes of this model that survivors remain immune to the virus, at least for the duration of the simulation, i.e. that they do not return to the susceptible population.

The Local Hospital

Some of the sick people choose to go to the local hospital for treatment. It is assumed that at the start of the epidemic nobody goes to hospital in the belief that the sickness will be treated in the local community. Only when it becomes apparent that the virus cannot be treated locally do people turn to the hospital for help.

TransferRateToHospital is defined as;

*SickInVillage * LocalAwarenessOfVirus * 0.5* (5)

where it is assumed that at most 50% of sick people go to hospital and then only when at least 5 local people have already died.[v]

Exploring Scenarios

As already stated, it is not the contention of the authors that the assumptions put forward in this paper are necessarily correct. Suppose for example, as might be the case in practice, that the infectiousness rate in the hospital is twice that in the local village. Figure 3[vi] shows the outcome of modifying the standard run to reflect this assumption. In this instance, the death rate in the local hospital increases to 450 people.

Note also that the death rate in the village also increases, to 35 people, even though the infectiousness rate in the village stays the same. This is because people continuously return home to the village after being treated for other illnesses and some of these have become infected during their stay in hospital.

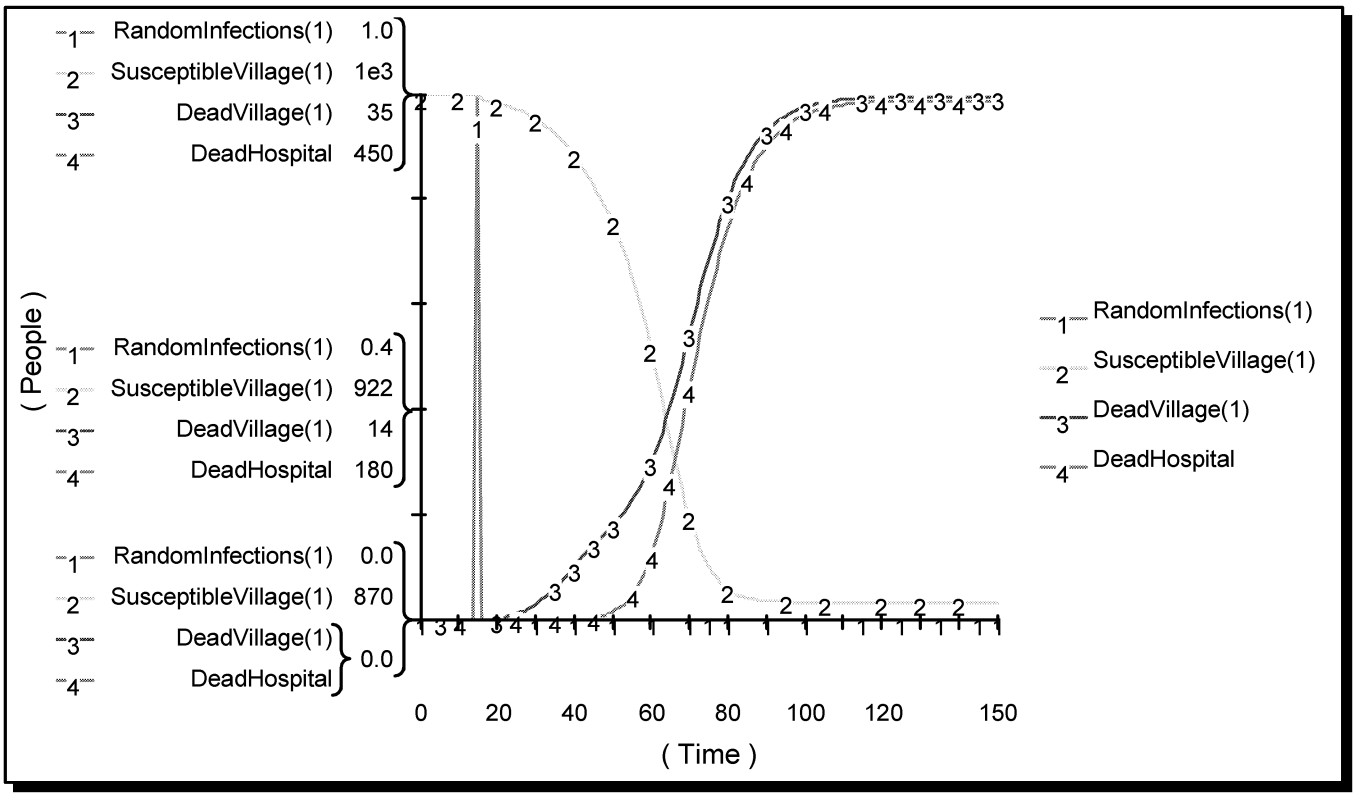


Figure 3: Higher infectiousness rate in local hospital

This last point raises a number of other possibilities. For example, the local hospital serves many villages. Patients can return home to the other villages infected with the virus, thus further spreading the epidemic. The array feature of the Powersim modelling software was used to extend the model to simulate five villages, each of which were served by the same local hospital. For this run, the hospital population was increased to 1000 to reflect the larger catchment area.

Now suppose that the virus mutates and the incubation period increases to 60 days. Under such a scenario, 1500 people would die in hospital over the period of 1 year (see Figure 4) and a further 240 deaths in the villages.

A higher incubation period for such a devastating virus as Ebola could destroy the entire local population were it not for outside intervention. The model could be further expanded to include a regional or national hospital sub-system but again, for simplicity, this was not attempted here.

Instead, *HospitalAwarenessOfVirus* (modelled on *LocalAwarenessOfVirus*), is used as a basis for forwarding specimens and clinical data to the regional or national laboratories and ultimately the CDC. There are a number of critical information delays in this process.

An information delay represents the process of a gradual, delayed adjustment of information moving towards the value being supplied by the source. The model considers four (cumulative) sources of delay, namely *DelayToAnalysisOfBloodSamples, DelayTo InvolveCDC, DelayInRecognisingEbola and DelayInEnforcement OfBarrierNursing.*

It is known from experience that it takes 5 days for the CDC to diagnose Ebola. Once diagnosed, it can take up to 1 week for CDC and WHO experts to arrive at the scene of the outbreak, locate all people exposed to the virus and enforce barrier nursing and encourage other protective measures. The delay to analysis of blood samples and the delay to involve the CDC are functions of the number of 'tiers' between the local hospital and the eventual diagnosis of the virus. In some African countries attempts have already been made to improve diagnosis facilities and, in such cases, there may be no need to involve the CDC. However, the standard run of this model assumes a delay of 5 days for each of these variables. The number of decisions to be made (the number of levels of authority to be called upon) impacts the nature of the delays in the system.

The standard run of the model assumes a total information delay of 22 days before barrier nursing was fully in place. Suppose for example that the virus could not be diagnosed or that the CDC did not get involved. In such circumstances, the initial regional population of 6,000 is reduced by 66%. Virtually everyone in hospital is dead. All that prevents total annihilation is the assumption that survivors remain immune.

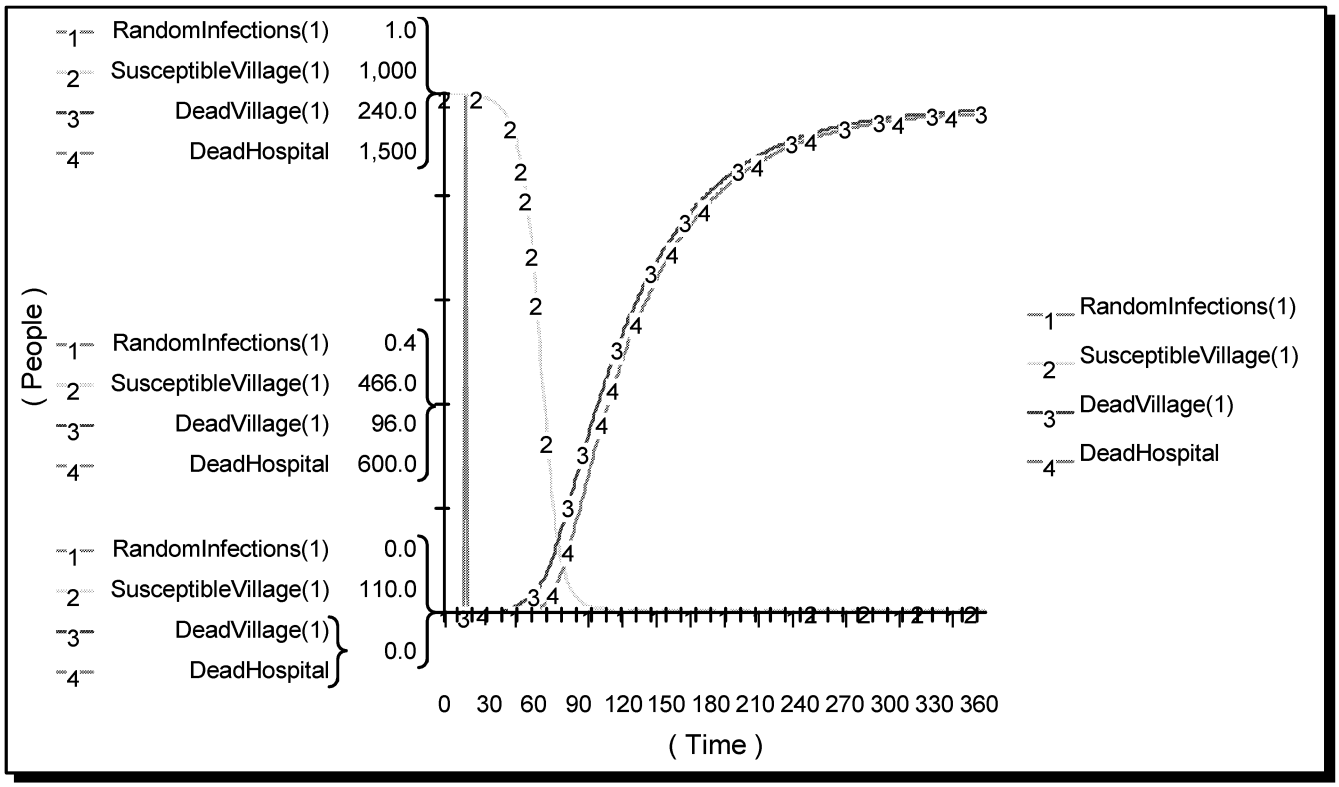


Figure 4: 60 day incubation period simulated over 12 months

Now assume that our understanding of the virus improves to the extent that Ebola is diagnosed in the local hospital by the time the 5th patient dies (a very unlikely scenario under current circumstances). Figure 5[vii] shows model output under this scenario.

The 5th person dies in the local hospital on day 51. Barrier nursing is enforced with immediate effect. A further 13 people still die in the hospital, even with the contact rate down to 0. This is because by the time 5 people have died in hospital, 6 are already sick and a further 4 are incubating. 6 people have died in the villages and a further 7 are incubating the disease by this time. The point here is that although national laboratory or CDC intervention is critical it is still incapable of preventing further deaths because of the delay between the 1st and 5th deaths.

Following on from this point, consider the scenario whereby the virus can be diagnosed in the local hospital once the first patient becomes sick. Under this scenario, the total death rate is restricted to 10 people (see Figure 6[viii])

CONCLUSIONS

"System dynamics models do not, of course, permit one to predict the future. Rather, the objective is experimental. It is to assemble the diverse bits of quantitative and qualitative information already available and evaluate which data are the most important in understanding the system as a whole." [Randers, 1973, p51]

The system dynamics model described in this paper is structured to represent a group of rural villages served by one local hospital, remote from regional or national laboratories. Such a structure typifies the circumstances of recent Ebola outbreaks in central Africa.

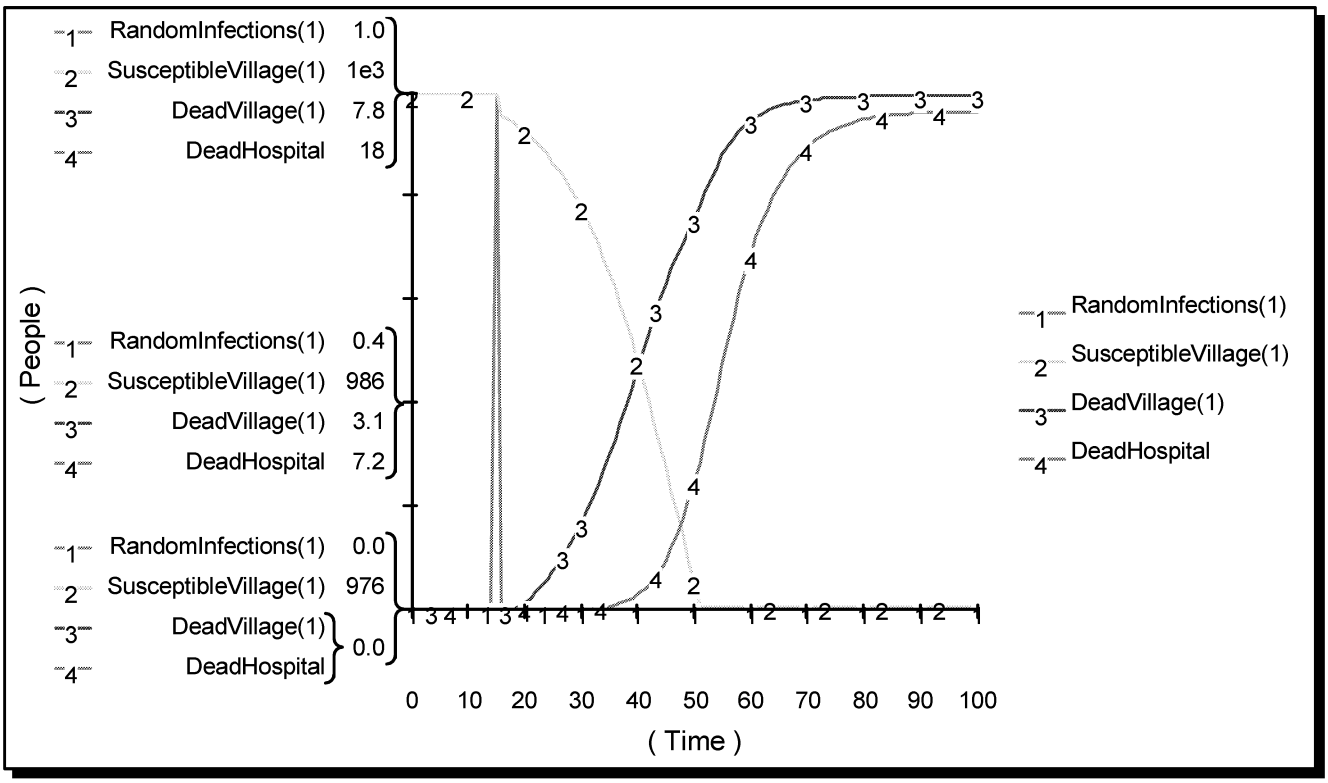


Figure 5: Diagnosis in local hospital

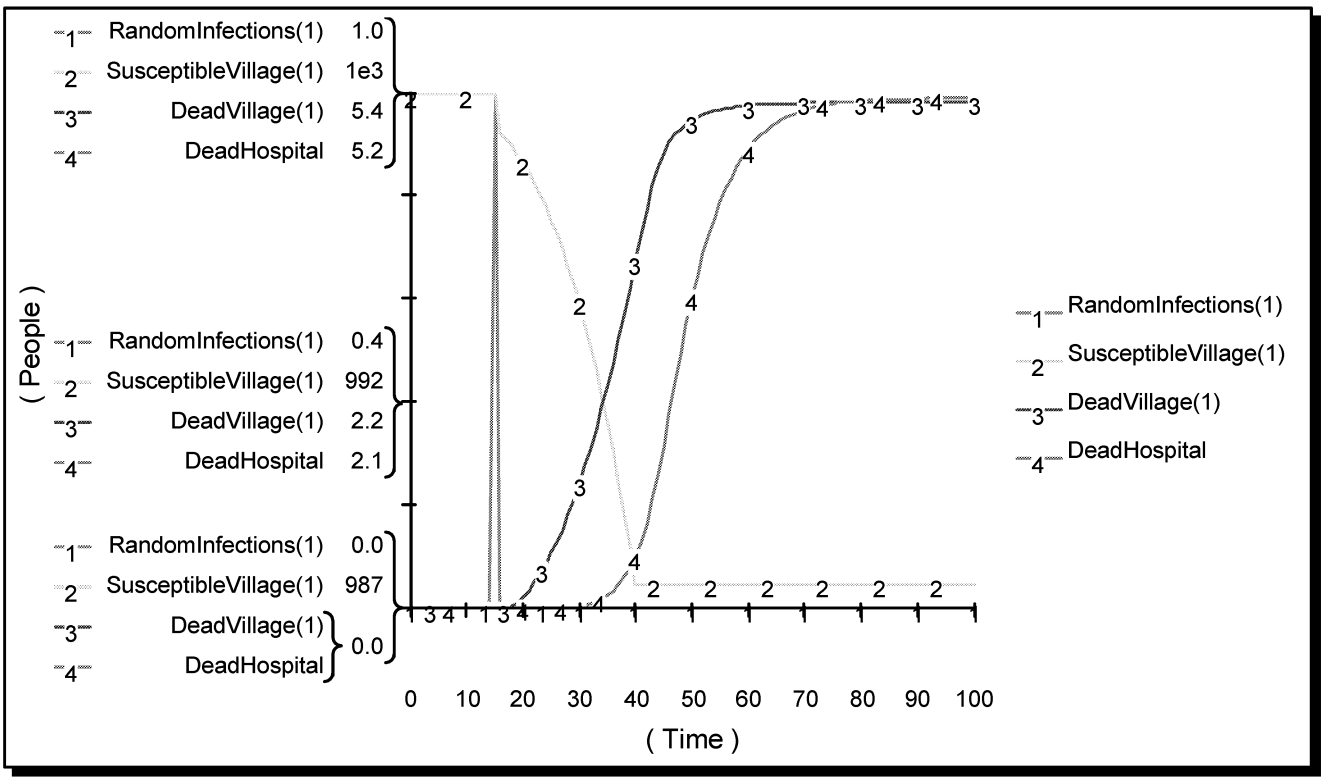


Figure 6: Diagnosis on 1st sickness in hospital

In creating the simulation model described here, the intention is to enrich existing verbal and written models of the factors impacting the spread of the Ebola virus. This model may be thought of as the synthesis of existing documentation to form a more transparent and dynamic tool for hypothesis testing. More specifically;

- Through a quantitative analysis of existing data, the model exposes, within the context of the problem area, the underlying assumptions used as a basis for system understanding. Through the compression of time, the model provides a means of taking these assumptions to their logical conclusions. Exposing assumptions in this way leaves less room for misinterpretation and provides a solid basis for enhancing the understanding of system structure.

- The behaviour-over-time graphs used to present model output provide a powerful means of exposing system complexity and, thus, increasing understanding.

- The model shows how the feedback mechanisms inherent in system structure influence behaviour patterns over time.

- The assumptions of the authors are made explicit through the structure of the model and the choice of parameter values. The behaviour-over-time graphs used to present model output provide a powerful means of exposing system complexity and, thus, increasing understanding. The exact nature of this complexity, as assumed by the model, is made explicit through the variable definitions. These definitions can be modified and the resulting changes in behaviour patterns examined.

This model, linked to a GIS, will provide an aid for health workers dealing with the Ebola virus and attempting to estimate the behaviour of a possible epidemic under a variety of situations. This is becoming increasingly important with the growth of travel to these remote regions and with the movement of refugees between countries (Kalipeni and Oppong, 1998, Freedman and Woodall, 1999).

Finally, the model aims to encapsulate best practice in the field of system dynamics. It emphasises the difference between actual and perceived conditions as a basis for action. It makes explicit the underlying assumptions as a basis for further expansion. It highlights system structure as a catalyst for change.

REFERENCES

Freedman, D.O., Woodall, J. 1999. "Emerging infectious diseases and risk to the traveler", *Medical Clinics of North America*, Vol. 83, No. 4, pp 865-883.

Kalipeni, E., Oppong, J. 1998. 'The refugee crisis in Africa and implications for health and disease: A political Ecology Approach', Social Science and Medicine, Vol. 46, No. 12, 15 June 1998, pp 1637-1653.

Le Guenno, B., Galabru, J. 1997. "Ebola Virus", *Bulletin de l'Institut Pasteur*, Vol. 95, No. 2, 6 April, pp 73-83.

Michie, C. 1999. "Lessons from the survivors of the Ebola infection", *Molecular Medicine Today*, Vol. 5, No. 7, pp 285.

Peters, C.J. 1997. "Ebola and Hanta Viruses", *FEMS Immunology and Medical Microbiology*, Vol. 18, No. 4, August, pp 281-289.

Randers, J., Meadows, D. 1973. "The Dynamics of Solid Waste Generation", *Towards Global Equilibrium: Collected Papers*, Meadows, D., Meadows, D. (Eds.), Wright-Allen Press, Cambridge, U.S.A., pp 165-212.

Shears, P. 2000. "Emerging and reemerging infections in Africa: the need for improved laboratory services and disease surveillance", *Microbes and Infection*, Vol. 2, No. 5, pp 489-495.

Sinha, G., Powell, C.S. 1996. "Shaking the Ebola tree", *Scientific American*, August, http://www.sciam.com/explorations/082696explorations.html.

Vick, K. 2000. Washington Post Foreign Service, October 14, pp A24.

SIMULATION OF CONTAGION BY TUBERCULOSIS IN PUBLIC PLACES AT US-MEXICO BORDER AREA

Carmen Jáuregui
Alfredo Cristóbal-Salas
Antonio Rodríguez-Díaz
Manuel Castañón-Puga
School of Chemistry Sciences and Engineering
University of Baja California (UABC) Tijuana, Mexico
{cjauregui, cristobal, ardiaz, puga}@uabc.mx

KEYWORDS
social simulation, scalable simulator, tuberculosis, infection study

ABSTRACT

In this paper we present a simulation of contagion by Tuberculosis at U.S.-Mexico border area. Tuberculosis (TB) is considered as one of the most highly infectious diseases. According to the World Health Organization, at least one third of the world population is infected. Recently, the number of TB cases has increased with the arousal of AIDS and aggravated by multidrug resistance TB mutants. TB could start an epidemic in this area under current conditions such as constant bi-directional local tourism, illegal immigration and extreme poverty in Mexican border cities. In this work, we analyze bares and discos at Tijuana during the weekend where Americans and Mexicans stay long time in closed and overcrowded areas. We also present details of our distributed scalable simulator based on object oriented paradigm.

INTRODUCTION

According to the World Health Organization (WHO), at least one third of the world population is infected by Tuberculosis (TB) and around 8.5 million new cases occur each year. TB is an infectious disease caused by some species of Mycobacterium, principally Mycobacterium tuberculosis (Schlossberg 1994). The infection by TB occurs in healthy individuals as a result of inhale a small number of TB bacteria (Daniel, 1991). Due to the number of bacteria exhaled by most of the infected individuals is small, usually it is required a long contact or areas with poor ventilation for an infection takes place. After infection, comes the primary TB which is hard to be detected (Castillo-Chávez et al. 1998). But, Individuals who have a correct and on time medication can become non-infectious; however, to meet these requirements it is necessary to involve hospitals, individuals and government. In this work we focus on the prevention rather on medication to content an epidemic by TB.

There are several works that relates TB and poverty (Bloch et al., 1989; CDC, 1991; Goldman et al., 1994; Bhatti et al., 1995; Styblo and Rouillon, 1981). Concentration of poverty in cities is the main factor that determines differences in the incidence rates. Inside urban areas, the cases vary widely, and they are bigger where poor people lives (Castillo-Chávez et al. 1998).

Particularly, in the U.S.-Mexico border there are several factors that elevate TB incidence and complicate case management. Some of these factors are: a) high TB rate in Mexico; b) low socioeconomic status in Mexican border cities and their limited access to health care; c) frequent bi-directional border crossings d) lack of coordinated efforts between both health jurisdictions and d) overcrowded living conditions in both sides of the border. Moreover, the presence of other diseases such as diabetes mellitus, lymphoma, cancer, and AIDS among others, contribute to the fast spread of TB (MMWR, 2001; SS, 2001). Therefore, any studies to control or decrease the number of cases of TB in U.S.-Mexico border area should be seriously considered especially because 95% of the total decease cases correspond to the economically active population (SS, 2001).

We believe prevention could be a solution to this problem. Showing how fast TB could spread under current conditions might make people take precautions. In (Castillo-Chávez et al. 1998) a mathematical model to study the dynamic of infection diseases is presented. However, this theoretical model might not be understandable by regular individuals. Therefore, we designed and implemented a simulator that matches current conditions at U.S.-Mexico border area. In (Jáuregui-Romo et al., 2003) we reported a basic centralized simulator architecture with one type of bacteria. In (Jáuregui-Romo et al., 2003-bis) we report the same simulator but now it makes difference between regular TB bacteria and mutants bacteria resistant to multidrug. Now, we present a distributed scalable simulator in which individuals can be infected with different types of TB bacteria. This simulator not only can be used in hospitals and government dependencies to analyze tendencies of epidemic but can also be used as educational tool to train individuals to detect, avoid or control an epidemic in this area. All details about this simulator are presented next.

The paper is organized as follows. In the next section, we present an overview of social simulation. The study subject model design, the simulator design and its features are presented in the following sections. Also, we present a TB spread in U.S.-Mexico border area case study. Lastly, conclusions and future work are presented.

SOCIAL SIMULATION

In recent years, social simulation has increased its popularity. First examples of computer simulation in the social sciences date from the 1960s; however, simulation has begun to be used widely since 1990s (Troitzsch, 1997). Simulation is now another way of doing research together with modeling and experimentation (Gilbert and Troitzsch 1999). Early work in computer simulation was focused on prediction, while social scientists tend to be more concerned with understanding and explanation (Gilbert and Troitzsch 1999).

Some of the purposes of simulation are to obtain a better understanding of some features of the social world, predict a particular event and to develop new tools to substitute human capabilities (Gilbert and Troitzsch 1999; Doran et al. 1995).

As computers and algorithms become more sophisticated, it is possible to model more complicated interactions between individuals and objects in real world. Nowadays, the object oriented programming (OOP) paradigm brings more facilities to model reality. Objects from the point of view of OOP are software bundles of related variables and methods. Software objects are often used to model real-world objects you find in everyday life.

OOP brings some advantages such as: fast development, easy maintaining, the re-use of designs and code, facilitate the programming of complex simulations, and allow the introduction of parallel programming techniques.

An object-oriented simulation (OOS) consists of a set of objects that interact with each other over time (Joines and Roberts, 1998). OOS refers to the construction of models using object oriented development and programming tools; where development means simulate entities in the real world and programming refers to the use of tools (libraries of classes) and data structures provided by programming languages.

STUDY SUBJECT MODEL

For the purpose of our simulation, all study subjects will be referred as Subjects (S). These subjects coexist in a virtual world that represents a common or public place. Every S is a tuple of six parameters S=<P, H, B, D, C, R> where

- P is the (x, y) position of S in the virtual world;
- H shows when an S is infected or not.
- B is a vector that stores the amount of bacteria inside of S. We make difference from reproduction and mutation of TB bacteria. When a bacterium reproduces the value in its particular position at B is incremented. When bacteria mutate, the amount of these new bacteria are stored in a new position at B. Each element of vector represents different type of bacteria which are resistant to different drugs or combination of drugs.
- D is a vector that stores the drugs' name which the bacteria are resistant. Each time bacteria are resistant to a new drug, the amount of bacteria is stored in B and the name of drugs is stored in D at the same position.
- C is the level of social consciousness or responsibility of not propagating the infection between S's. It is measured as percentage. Levels near to 0% mean a not careful, not educated, or not instructed S while values near to 100% revels levels of responsibility, consciousness or training. This parameter increases each time S's attend to training courses. We do not consider the case when S's may attend to training courses but they do not desire what they learned. We leave it as future work.
- R is associated to the S's immune system and it means the capability to resist TB. It is also measured as percentage. When R is near to 0% then S is susceptible to TB however, if R is near to 100% then S is resistant to any TB bacteria. This value can be incremented with positive events such as: balance meals, exercise, use of vitamins, or having medical assistance. It can also be reduced by negative events such as: drinking, smoking, etc. These factors are produced randomly during our simulation. R also decreases with the absence of events, for instance, if no event at all is generated then R decreases automatically.

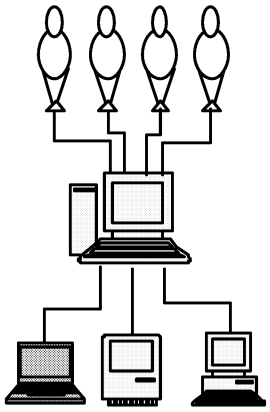


Figure 1: Scalable Simulator Design

SIMULATOR DESIGN

In previous work (Jáuregui-Romo et al., 2003; Jáuregui-Romo et al., 2003-bis) we introduced a centralized multi-threaded simulator to handle the virtual world. Now we present a scalable distributed simulator under the message-passing paradigm. Our simulator consists of three main components (see figure 1 for details): A set of virtual world administrators (VWA's), a set of Subjects (S's) and a set of displays (GD's). The functionality is presented next.

- A VWA manages matrix of integers that represent the map of the virtual world. The matrix contains one of the following values, zero that represents the floor, one that represent a portion of a wall and two that represent a S. Due to in social simulation we pretend to observe several individuals, the communication between the S's and the VWA can become a bottleneck. That is why, in our simulator design it is possible to have more than one VWA managing different sectors of a map.

For instance, a two floor building can be managed by two VWA's, one for each floor or if population grows more than a threshold then the actual map can be split and different VWA's can manage a part of it. WWA's have three main responsibilities

1) Each one of the VWA has to broadcast the map of the virtual world at the beginning of simulation to GD's that request it.
2) During the simulation, each VWA has to send the current position of each S to GD's that request it.
3) Validate each movement of S inside the virtual world. For instance, S cannot walk through a wall.

- The set of GD's show in screen what is happening in the virtual world. Each one of them receives from the VWA the map and vector of positions and draws them adding the corresponding textures. There is no communication between GD's or between GD's and S's. A GD only communicates with just one VWA.
- The set of S's represent virtual persons. S's has set of predefined paths to move inside the virtual world. It randomly chooses one of them and each time it wants to move, a communication with VWA needs to be done to validate the movement. S's only communicates with the VWA this means that communication between S's only is possible trough a VWA. When a S reaches the end of the virtual world managed by a particular VWA, then the VWA send the address of the new VWA and the S then start communication with the new VWA. For instance, in a simulation of a two floors building, when the S_1 reaches the elevator in the first floor then the VWA managing the first floor sends to the S_1 the address of the VWA managing the second floor.

FEATURES OF SIMULATOR

Features of our simulator can be summarized as follow.

- Each one of the components of our simulator can be run transparently in a separated computer, i.e., it was designed for a distributed memory parallel computer.
- Due to each component can run in different computer then the entire system is scalable.
- Communication between components is performed directly by sockets and the content of each message is text-only so our simulator can be multi-platform. Any component can run in different operative systems, type of computers (Desktop, workstation or even a supercomputer). An interesting configuration that show the functionality of our design could be the following, run the VWA in a workstation, the S's in a cluster of PC's and supervise the simulation with a GD in a PDA. This is our current work.
- Due to each S can be a completely separated process in the system. It is possible to perform complicated computations that can be dedicated to reasoning, learning or complex storage of information.
- It is possible to generate different kinds of GD's. Current implementation of our simulator has a 2D GD but we are working on a 3D GD.
- Each component can be programmed separately in different programming languages without modifying the entire system.
- Each component is modeled under the object oriented paradigm. More information of this model can be obtained from (Jáuregui-Romo et al., 2003).

TB SPREAD IN U.S.-MEXICO BORDER AREA CASE STUDY

In U.S.-Mexico border area there is a well known phenomenon, people from both countries cross the border during the weekend to spend time in the opposite country. Young Americans spend much of their time in small stores and discos in Mexico and Mexicans spend much of their time in swap-meets and restaurants in U.S. According to (Raffalli et al., 1996) there are several factors that are essential for spreading of TB: a large number of individuals that are susceptible to TB, an individual that spread TB, overcrowded rooms and lack of ventilation. These conditions are present in all public places mentioned before.

Figure 2 shows possible configurations of infected and not infected study subjects that match conditions observed in U.S.-Mexico border area. We test our simulator with different scenarios to compare speed of TB spread. The first scenario we introduce a population of individuals S_1. This scenario could represent the current conditions in the border area. In these circumstances appears an individual S_3 that interacts in the virtual world. We consider the time needed to infect the entire population. This time is the fastest spread of TB. This experiment shows the possible consequences that could occur in case that TB or any other highly infectious disease could appear in the border area population.

$S_1 = <(x_1, y_1)$, not infected, empty B, empty D, low C, low R>
$S_2 = <(x_2, y_2)$, not infected, empty B, empty D, low C, high R>
$S_3 = <(x_3, y_3)$, not infected, empty B, empty D, high C, low R>
$S_4 = <(x_4, y_4)$, not infected, empty B, empty D, high C, high R>

(a)

$S_5 = <(x_5, y_5)$, infected, not empty B, not empty D, low C, low R>
$S_6 = <(x_6, y_6)$, infected, not empty B, not empty D, low C, high R>
$S_7 = <(x_7, y_7)$, infected, not empty B, not empty D, high C, low R>
$S_8 = <(x_8, y_8)$, infected, not empty B, not empty D, high C, high R>

(b)

Figure 2: Possible Configurations of (a) Not Infected and (b) Infected Study Subjects with Different Levels of Resistance to TB Bacteria and Social Consciousness

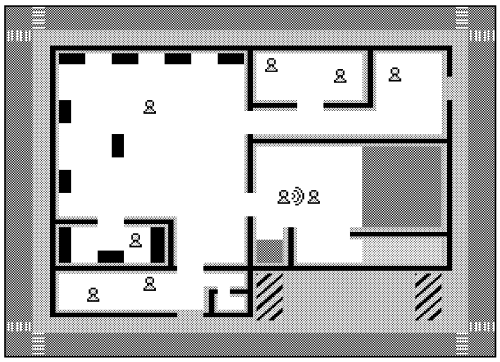


Figure 3: Screen-Shot of a 2D Display

In a second experiment we take the previous population, i.e. individuals S_1 and a single individual S_5 but now the VWA sends an alert signal to the population announcing a possible epidemic. All individuals near to the infected one start to increase the immune system by generating positive events such as: balanced meals, exercise, avoid smoking, etc. This social security policy intents to create a wall to avoid propagation to the rest of population. Thus, individuals S_1 now will have configuration S_2. A similar effect is obtained simulating a prevention of highly contagious campaign promoted by health dependencies. This campaign could intent to increase the population social consciousness. Then individuals S_1 now would have configuration of individuals S_3. In both cases we see a slower propagation and this could give to health dependencies more time to apply other plans to control the spread.

In a third experiment we have original configuration (several individuals S_1 and a single individual S_5) but now the VWA send an alert signal to S_5 instead to the rest of population. By applying a plan to increase its immune system then infected individual (S_5) would have the configuration of individuals S_6. This plan can be easily applied by providing vitamins or nutritional supplements to infected individual. Also, it is possible to provide information to the infected individual such a courses or sessions to train him/her to avoid propagation. Then, infected individual (S_5) would have configuration S_7. In both cases simulation shows a slower propagation. The difference with previous scenario is that the previous one could provoke panic in the population which could make a worst scene.

Obviously, TB spread is controlled is individuals S_4 and S_8 are present in the simulator. This is the fourth scenario we test in our simulator. It is important to mention that even the individuals' financial status could be so critical that R levels cannot be increased, as happen in Mexican border area cities, it is possible to increase the social consciousness and this could help to control de TB spread. Figure 3 shows a screen-shot of a 2D GD of our simulator.

CONCLUSIONS AND FUTURE WORK

In this paper we present a distributed scalable simulation to analyze the spread of TB in U.S.-Mexico border area. This simulator helps to a better understanding of population behavior, reducing the study times and discovering strategies for problem solving, specially on infectious diseases involving etiological agents with duplication times as long as 72 hours or more, as seen on Mycobacterium strains. The contagion by TB in U.S.-Mexico border might arise due to the increasing of sickness such as AIDS or SARS which reduce the resistance of immune system to bacteria or viruses. Both governments have implemented health plans to control the propagation of TB but these health plans are not enough to control TB at the border line. We believe that prevention is a key issue; however, the lack of information among inhabitants and the extreme poverty at Mexican border towns make the TB spread a major problem. Our simulation shows the advantages of dropping bad habits and improving the way of life, but also shows the importance of both governments taking precaution plans to control TB spread along the common border.

Due to the importance of this work, extensions of this simulation are considered. We plan to simulate propagation of high risk infections in working places and their impact on productivity.

ACKNOWLEDGMENTS

The authors take pleasure in acknowledging Reyes Amador Soto Higuera, Israel Cabrera, Manuel Arturo Márquez Borbón and Israel Ochoa for their participation in the experimental analysis. This work is partly supported by UABC (Unversidad Autónoma de Baja California) under grants #364, #313, and #371 and by SEP (Secretaría de Educación Pública) under PROMEP support for research projects. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the UABC or of SEP.

REFERENCES

Bhatti N, Law MR, Morris JK, Halliway R, Moore-Gillon J. Increasing incidence of tuberculosis in England and Wales: a study of the likely cases, BMJ 1995; 310: 967-969

Bloch AB, Rieder HL, Kelly GD, Cauthen GM, Hayden CH, Snider DE Jr. The epidemiology of tuberculosis in the United States, Clin Chest Med. 1989; 10:297-313

Castillo-Chávez C., Capurro AF, Velasco-Hernández J., Zellner ML. El transporte público y la dinámica de la tuberculosis a nivel poblacional. Aportaciones matemáticas Serie comunicaciones vol. 22 (1998) pp. 209-225.

CDC (Center for Diseases Control), Tuberculosis – Buffalo, New York. MMWE Morbidity and Mortality Weekly Report 1967; 16:1995

Daniel TM. Tuberculosis In Wilson JD, Braunwald E, Isselbacher KJ, Petersdorf RG, Martin JB, Fauci AS, Root RK (Eds). Harrison's principles of Internal Medicine, 12th Ed, McGraw-Hill, Inc, New York, NY, 1991, pp. 637-645

Doran, J.; N. Gilbert; U. Mueller; and K.G. Troitzsch. 1995. "Social Science Microsimulation: A Challenge to Computer Science". Dagstuhl-Seminar-Report 112. D-66687 Wadern, Germany.

Gilbert, N., and K.G. Troitzsch. 1999. "Simulation for the Social Scientist". Open University Press ISBN 0-335 19745-0

Goldman JM, Teale C, Cundall DB, Pearson SB. Childhood tuberculosis in Leeds, 1982-90.: Social and ethnic factors and the role of the contact clinic in diagnosis. Thorax 1994; 49:184-185.

Jáuregui-Romo C, Cristóbal-Salas A., Rodríguez-Días A., Castañon-Puga M. Modelo de Objetos para la Simulación de Contagio por Tuberculosis en la Frontera México-E.U.A. to appear in proceedings of the XVI Congreso Nacional y II Congreso Internacional de Informática y Computación de la ANIEI (CNyCIIC-ANIEI), October 2003

Jáuregui-Romo C, Cristóbal-Salas A., Rodríguez-Días A., Castañon-Puga M. (bis) Simulación de contagio y farmacoresistencia a la terapia antituberculosa to appear in the proceedings of the XXXVI Congreso Nacional de Ciencias Farmacéuticas 2003, November 2-6, 2003 Cancún Mexico.

Joines, J.A. and S.D. Roberts. "Fundamentals of object-oriented simulation". In Proceedings of the 30th conference on Winter simulation 1998 , Washington, D.C., United States Pages: 141 - 150 1998 IEEE Computer Society Press ISBN:0-7803-5134-7

MMWR (Morbidity and Mortality Weekly Report). January 19, 2001/ 50(RR1);1-2. Centers for Disease Control and Prevention. USA.

Raffalli J, Sepkowitz KA, Armstrong D. Community-based outbreaks of Tuberculosis. Archives of Internal medicine 1996; 156:1053-1060

Schlossberg D (1994). *Tuberculosis*. Third Ed. Springer-Verlag. USA.

SS (Secretaría de Salud). *Decreto por el que se aprueba el Programa Nacional de Salud* 2001-2006. Diario Oficial de la Federación, México, 21 de Septiembre de 2001. Tomo DLXXVI; Num 15: pp 1-128.

Styblo K, Rouillon A. Tuberculosis in the World. Estimated global incidence of smear-positive pulmonary tuberculosis. Unreliability of officially reported figures on tuberculosis. Bull Int Union Tuberc (1981); 56: 118-126.

Troitzsch K.G., 1997. "Social Science Simulation - Origins, Prospects, Purposes". R. Conte, R. Hegselmann, P. Terna, editors.: "Simulating Social Phenomena", Berlin (Springer) (LNEMS 456), 41-54

SIMULATION OF ECOSYSTEMS

CHANGING THE LEVEL OF DESCRIPTION IN ECOSYSTEM MODELS: AN OVERVIEW

Pierrick Tranouez - Sylvain Lerebourg - Cyrille Bertelle - Damien Olivier
Laboratoire d'Informatique du Havre
Faculté des sciences et techniques
Université du Havre
25, rue Philippe Lebon
BP 540 76058 LE HAVRE Cedex
E-mail : Pierrick.Tranouez@univ-lehavre.fr

KEYWORDS

aquatic ecosystems, multiscale, simulation, organizations detection

ABSTRACT

Aquatic ecosystems are natural open holarchic systems. They are crossed by energy fluxes that give them structure. One of these fluxes is mass transport, which is carried by the fluid flow. This flow is the essential vector for interaction inside the system and thus one of the main contributors to emergent formations structuring these ecosystems. Furthermore, to follow ecosystem evolution, because of their holarchy, it is necessary to observe and represent organizations that span over different scales. We finally present a fluid flow simulation which dynamically detect emergent formations, then manage them on different scales.

INTRODUCTION

France Ministère de la culture has been fighting an increasingly difficult battle against the Lascaux cave since 1963. Discovered in 1940, this rich item of our cultural heritage was opened to the public in 1948. It would then be closed in 1963 because of the proliferation of algae and their bacteria on the painted walls. Since then, the commission in charge of the preservation of the cave has tried to bring it back to a steady biotic state akin to what it was before the discovery of the cave. So far they haven't succeeded, as each solution to one problem was the cause of the next. For instance, the formol used to get rid of the algae was the food used by a *fusarium* mushroom to colonize the caves. Each time they tried to alter a factor to right a perceived wrong, the ecosystem of the cave adopted a new unforeseen trajectory.

This is an example of why ecosystems are tough to handle scientifically. By their size and complexity, they resist reductionist approaches. Ecosystem theories are near impossible to prove formally, as is often the case in life science, and experiments range from very difficult to impossible to carry upon them, because of both structural and moral problems.

Simulations remain therefore among the best tools to validate the models specialists of ecosystems create. Nonetheless, the models are necessarily huge and complex, and computer science must help provide said specialists with tools adapted to running this kind of simulation. This is what we are going to try to describe here.

First, we will describe how, from the original notions of ecosystems, we have now reached a description using advances in systemic and thermodynamics, and how this holarchic description, in the meaning of Koestler, guides the way we must simulate things.

We will then describe how these multiple scales are handled in other parts of science and simulation, in both global and local approaches to modeling.

We will conclude with describing another way of handling scale transfer and an application to the simulation of a fluid flow in an estuarial ecosystem.

REASONS FOR CHANGING THE SCALE IN A SIMULATION OF AN ECOSYSTEM

Ecosystems are ecological systems.

From ecology to ecosystems

The British botanist Tansley coined the word ecosystemks in 1935. This is how he defined them: "The more fundamental conception is … the whole system (in the sense of physics) including not only the organism-complex but also the whole complex of physical factors forming what we call the environment. We cannot separate them (the organisms) from their special environment with which they form one physical system ...It is the system so formed which [provides] the basic units of nature on the face of the earth … There ecosystems, as we may call them, are of the most various kinds and sizes."

Following this line of thought, ecosystems are often roughly described by the synthetic equation: "Ecosystem = Biotope + Biocoenosis". They are therefore tackled through a reductionist approach, as is common in science. In this method, each part of the subject of the study is divided in smaller parts, then again until it is estimated that dividing it further wouldn't provide anymore simplicity or clarity. Then, a holistic approach of each part is taken, trying to put back together everything divided during the analysis.

This method works for instance in many sub-domains of physics. However, it does not, work well for ecosystems.

One problem comes from the synthetic, holist part, which proves too vast to handle. But also the first part, the reductionist analysis, through its simplifying way that is at the core of its principle, "breaks" what it studies when it divides it. Unlike massive systems whose division gives more systems whose sum of the masses gives the original mass, ecosystems are more akin to living systems, where dividing leads to unliving systems, not to smaller live ones. The coupling between the different scales is just not subtle enough to represent in an appropriate way what exists in ecosystems.

Event though Tansley used the word "system" in its thermodynamic acceptation, we will now see how this domain alone, if necessary to describe ecosystems, is not sufficient. We must further develop it thanks to Von Bertalanffy systems.

Ecosystems as "General System Theory systems"
Von Bertalanffy introduced a new meta-scientific discipline called General System Theory (Von Bertalanffy 68). This method has later been developed, refined and used by such as Le Moigne (Le Moigne 94) or, as for the domain of our interest, the Odum brothers (Odum and Odum 53).

A system in General System Theory is a set of interacting elements that verify three principles (Frontier and Pichot-Viale 98):

- *Principle of mutual dependence.* At least some of the structures and dynamics of the elements of the system depend one from another. As a consequence, if you isolate one of these elements, you modify it, and if you act upon one of them, you influence some others
- *Principle of an emerging entity interacting with its environment.* From the interaction of some of its elements emerge a "new" entity, which differs from its components by its structure, relations and dynamic.
- *Principle of a retroaction of the emerged entity on its components.* The set modifies the way its composing elements behave; this principle can somewhat be seen as a specialization of the first principle.

Following this definition, ecosystems are "General System Theory systems".

Ecosystems as thermodynamic system
In thermodynamics, systems are parts of matter singled out from their surroundings. These surroundings are the rest of space around the singled parts. Material systemic systems are therefore thermodynamic systems. Ecosystems are always in part material. They therefore incorporate subsystems that are thermodynamic.

When our focus of study of ecosystems is precisely its thermodynamic part, we may abusively call the ecosystem a thermodynamic system, thus neglecting for instance the information relations inside the ecosystem. This is the way it is usually done, as in [Jorgensen et al. 00]. This practice works fine as long as the study is kept to structures little influenced by entities with perception like animals, whose interactions are poorly handled by thermodynamics.

Ecosystems are crossed by flows of matter and energy. If some of these flows are entirely internal to the system, some of them originate from or continue into the environment of the system. Ecosystems are therefore open thermodynamics system.

Furthermore, their trajectory seldom includes equilibrium states, unless the ecosystem is in decline (Frontier and Pichod-Viale 98). The flows of matter and energy may lead them to oscillate around stable states: ecosystems are dissipative systems.

Ecosystems are SOHOs
Ecosystems are therefore what Koestler calls SOHOs, for Self-Organized Holarchic Open systems (Koestler and Smythies 69). A holarchy is an extension of the notion of hierarchy, where the top/down influence is not privileged. Each member of the holarchy is called a holon.

A model of an ecosystem must consequently include this SOHO aspect. Its holarchic part will lead to a multi-level features for the model, while the openness will put an emphasis to the modeling of flows.

Simulation of complex systems

Complexity vs. reductionism
As explained in (Adami 98), to study how mass works in a material system, dividing this system into smaller parts is a good method. Indeed, each of his subsystems is massive, and therefore the study, the reductionism, can continue.

This is not so with living systems. If you divide a living system into smaller parts, the odds are good that all you reap is a heap of dead things. That's because the life question of a live system is *complex*. This means that what is important is not so much the parts of the systems, nor the parts of these parts, but the functioning relations that exist between them.

This is one of the two main reasons why one may want to integrate the multiple possible scales of description into a simulation. When you enquire about a complex question in a system, you need to choose carefully the needed levels of description, as you can't simplify them. Furthermore, these needed levels may change during the simulation, and it would be a fine thing if the simulation could adapt to these variations.

Thus changing the scales of description during the simulation could be useful for the accuracy of the answers to complex questions regarding the system the simulation may provide. And then there is the understanding of these answers.

Clarity of the simulation
Users of a simulation *question* it. Final users ponder about the future of the thing simulated in various circumstances, developers try to ascertain the validity of their model and of its implementation, but all use it with a purpose in mind.

Choosing the right level of description is then important to give a useful answer. If the simulation is able to adapt its descriptions to what is needed by its user, lowering the

noise and strengthening the signal, by choosing the right level(s) of description, it will be a better tool. For example in our application, a simulation of a fluid flow in an ecosystem, this help takes the form of hiding tiny perturbation and putting forward the main structures of the flow that emerged during the simulation

METHODS FOR CHANGING THE SCALE IN A SIMULATION

Law-based vs. rule-based models

Classifying the various ways science can tackle problems is an arduous task. We will nonetheless distinguish two rough categories of models.

Law-based models are the most used in science, most notably in physics. They are often continuous, especially in their handling of time and space, and based on a differential formulation whose resolution, ideally formal but often numerical, computes the values of state variables that describe the studied domain. Those methods are sometimes also called *global* or *analytical.*

In rule-based models, the studied domain is discretized in a number of entities whose variations are computed through the use of rules. There is therefore no longer a global description of the domain, nor is there a priori continuity. Cellular automata fall in this category of course, and so do objects/actors/agents. Those models have had a strong influence on game theory, and from there directly on social models, and later on other domain through computer science for instance, at least by way of metaphors. Those models have other names depending on the domain where they are used, ranging from *micro-analytical* in sociology, to *individual-based* in life sciences or just simply *local.*

Both kinds of models can be deterministic or stochastic. Finally, so as to blur the distinctions a bit more, models may include sub-parts falling in any of these categories. This is often the case with ecosystems for instance.

Changing the scale in law-based models

Accessing different levels of description in these models is often done through integration. Indeed, as said before, state function in these models are often continuous, and can therefore be integrated. New state functions are then valued or even built, on another domain and based on different phenomenological equations. For example A. Bourgeat (Bourgeat 97) describes fluid flows in porous milieus, where, from Navier-Stokes equations, through integration and the addition of an extra parameter, he builds a Darcy law. These changes of equations description from one level to another alter sometimes drastically the linearity of the models and may lead to the introduction of new parameters that act as a memory of the local domain inside the global one.

In a similar way to this example, the change of level of description in analytical models is often performed *a priori*, at the building of the model.

Changing the scale in rule-based models

Models based on rules offer a wider variety of ways of changing the levels of description. Indeed, local approaches are better designed to integrate particularities of very different entities and their mutual influence, as is the case when entities of various scales interact.

Cellular automata
The first individual based computer science structures may have been cellular automata. If they were created by Stanislas Ulam, Von Neumann self-replicating automata may have been the foundation of their success. Ulam himself already noticed that complex geometric shape could appear starting with only simple basic blocks. Von Neumann then Langton (Langton 86) expanded this work with self-replicating automata.

If shapes and structures did appear in the course of these programs, it must be emphasized that it were users, and not the programs themselves, that perceive them. Crutchfield (Crutchfield 92) aimed at correcting that trend, by automating the detection of emergent structures.

Detecting structures has therefore been tried, but reifying these structures, meaning automatically creating entities in the program that represent the detected structures has not been tackled yet, as far as cellular automata are concerned. It could be that the constraint on its geometry and the inherent isotropy of the cellular automata are in this case a weakness.

Ecology
Since the beginning of the use of individual based models in ecology, the problem of handling the interactions between individuals and populations occurred (De Angelis and Gross 92). The information transfers between individual was handled either statistically (Caswell and John 92) or through the computing of action potential (Palmer 92).

DAI uses.
Most software architectures designed to handle multiple levels of description are themselves hierarchical. They often have two levels, one fine grained and the other coarse grained. Communication between these two levels could be called decomposition and recomposition, as in (Marcenac 97).

In 1998, members of the RIVAGE project remarked in (Servat et al. 98) that it was necessary in multi-agent simulations, to handle the emergent organizations, by associating them with behaviors computed by the simulation. Before that, were handled only border interactions between entities and groups (Gasser 92).

This led in D. Servat PhD thesis to a hydrodynamic model incorporating in part these notions. In his Rivage application, water bowls individuals are able to aggregate in pools and rivulets. The individuals still exist in the bigger entities. The pros are that it enables their easily leaving the groups, the cons that it doesn't lighten in any way the burden of computing. Furthermore, these groups do not, to our knowledge, have any impact on the trajectories of the water bowls.

APPLICATION TO THE FLUID FLOW OF AN ESTUARIAL ECOSYSTEM.

Ontological summary

The fluid flows that constitute the ocean currents on the planet are the result of an important number of vortexes of different scales. Turbulent movement can also be decomposed into vortexes, on scales going down to the near molecular. Viscosity then dissipates kinetic energy thus stopping the downward fractal aspect of these vortexes (Lesieur 87). There are qualitatively important transfers of energy between these various scales of so different characteristic length. Representing these is a problem in classic modeling approaches.

In classic, law based models, turbulent flows are described as a sum of a deterministic mean flow and of a fluctuating, probabilistic flow. These equations (Navier-Stokes) are not linear, and space-time correlation terms must be introduced to compensate for that. These terms prevent any follow up of the turbulent terms, and thus of the energy they transmit from one level to another.

A pure law based approach is therefore not capable of a qualitative analysis of the transfer of energy between the different scales of a turbulent flow. A multi-level model, where multiple scales of vortexes would exist, and where they would be able to interact, would be a step in this qualitative direction.

Treatment of multiple scales

Fluid mechanic model and its structures
There are a number of models used to describe fluid flows. The set we use here are based on a discretisation of the flow, and are called vortex methods (Leonard 80).

In vortex methods, the flow is separated in a number of abstract particles, each being a local descriptor of the flow. These particles indicate the speed, vorticity etc… of the flow where they are located.

These particles are not fixed: they are conveyed by the fluid they describe.

This model is of interest to us as it is a local model, hence better able to deal with local heterogeneities. The values of the properties the particles describe are computed through the interactions between the particles, most notably through Biot-Savart formula. More details on this computation can be found in (Bertelle et al. 00).

The vortex method we use is of $O(n^2)$ complexity. Finding ways of lightening this calculus is therefore important. One lead is through making our model multi-scale, and only computing entities at the scale we need them. This is our second motivation for our using different levels of description.

In order to have different levels of description, we will have to use an adapted description of the simulation entities.

These entities come and go during the simulation, and thus we need a method to change the level of their description *during* the simulation, and not beforehand the way it is usually done.

In our fluid flow, the main entities as we explained are vortexes. Not only do we therefore need to detect emerging vortexes by monitoring lower level vortexes particles, but also, as these vortexes aggregate among themselves to form even bigger vortexes, make this detection process iterative.

Detecting the structures is not enough: we also need to create them in the simulation once they are detected. We must make these new entities live in the simulation, interacting with its various inhabitants (most notably particles, vortexes). They must evolve, whether it is growing or decaying to its possible disintegration.

Let us now describe our recursive detection-creation–evolution-destruction cycle.

Detecting emergent vortexes among the vortex particles
Structures are detected as clusters of particles sharing some properties. For vortexes these properties are spatial coordinates and rotation sense.

As described in the following figure, the process is:

- Delaunay triangulation of the particles
- Computation of a minimal spanning tree of this triangulation
- Edges that are too much longer than the average length of edges leading to the particles are removed. So are edges linking particles of opposite rotational.
- The convex hull of the remaining trees is computed
- An ellipse approximates the hull through a least square method

Further details on this process can be found in (Tranouez et al. 01).

Scale transfer : making simulation entities of the detected structures
Detected structures are created in the simulation where they take the place of the particles whose interactions gave them birth.

The vortex structures are implemented through multiplicity automata (Bertelle et al. 01). These automata handle both the relations between higher level vortexes and the relations between them and the basic particles.

The relations between vortexes and their environment are handled through a method based on the eco-resolution model (Drogoul et al. 92), in which entities are described through a perception and combat metaphor. The associated perceptions and actions are:

- Perceiving an intruder means being on a collision course with another vortex.
- Attacking another vortex means sending it a message.
- Being attacked means receiving such a message.

- Fleeing means being destabilized: the vortex structure shrinks and creates particles on its border. Too much flight can lead to the death of the structure, which is then decomposed in its basic particles.
- Getting satisfaction means aggregating surrounding particles of compatible vorticity. This calculation is done through a method close to the initial structure detection: Delaunay triangulation, spanning tree, removal of edges. Compacity criteria are then used to estimate whether the tree should be added to the vortex and thus a new ellipse be computed or not. For instance in figure 1, the particles on the lower left will be aggregated while those on top won't.

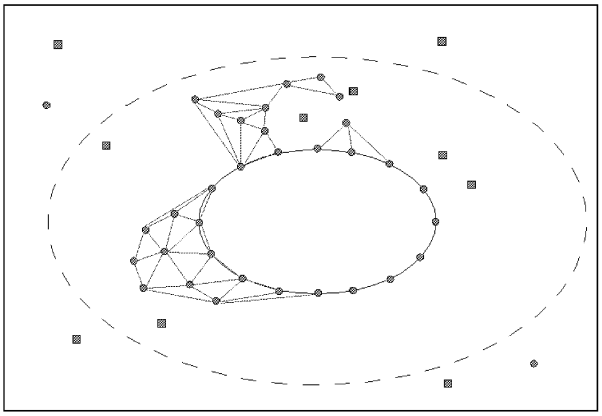


Figure 1 : to aggregate or not to aggregate

The described process is then iterated. New structures are detected and implemented, while others grow, shrink or disappear altogether.

CONCLUSION

A promising approach of ecosystem modeling nowadays is through their representation by holarchic thermodynamic dissipative systems.

The thermodynamic side of this description imposes an appropriate handling of flows in the ecosystem. The holarchic side imposes a software model able to handle multiple level of description, not only in their existence but also in their functioning, which means detecting and managing structures that may appear during the simulation.

The detection part has been attacked for the past 10 years, but the ensuing simulation of the structures is not so densely described.

We propose an essentially rule-based model, with a Navier Stokes law-based bottom foundation that has these properties in our simulation. It is also infinitely recursive, and not limited to just two levels. This is possible because of the fractal nature of the model used to describe the fluid flow.

REFERENCES

Bertelle C., D. Olivier., V. Jay., P. Tranouez.and A. Cardon., « A multi-agent system integrating vortex methods for fluid flow computation », *16th IMACS Congress2000*, vol. 122-3, Lausanne (Switzerland), August 21-25 2000, electronic edition.

Bertelle C., M. Flouret., V. Jay., D. Olivier and J-L Ponty., « Automata with multiplicities as behaviour model in multi-agent simulations », *SCI'2001*, Orlando (USA), 22-25th July 2001.

Bourgeat A., « Quelques problèmes de changement d'échelle pour la modélisation des écoulements souterrains complexes », BLASCO F., Ed., *Tendances nouvelles en modélisation pour l'environnement*, Elsevier, 1997, p. 207-213.

Caswell H.and A. John., « From the individual to the population in demographic models », DEANGELIS D., GROSS L., Eds., *Individual-based models and approaches in ecology*, Chapman et Hall, 1992, p. 36-66.

Crutchfield J., « Discovering Coherent Structures in Nonlinear Spatial Systems», BRANDT A., RAMBERG S., SHLESINGERM., Eds., *Nonlinear Dynamics of Ocean Waves*, Singapore, 1992, World Scientific, p. 190-216.

Deangelis D.and L. Gross., Eds., *Individual-based models and approaches in ecology*, Chapman et Hall, 1992.

Drogoul A.and C. Dubreuil., « Eco-Problem-Solving : results of the N-Puzzle », DEMAZEAU Y.,WERNER E., Eds., *Decentralized Artificial Intelligence III*, North Holland, 1992, p. 283-295.

Frontier S.and D. Pichod-Viale., *Ecosystèmes*, Dunod, 1998.

Gasser L.., « Boundaries, Identity and Agggregation : Pluralities issues in Multi-Agent Systems », WERNER E., DEMAZEAU Y., Eds., *Decentralized A.I.*, Elsevier, 1992.

Jorgensen S.and F. Müller, *Handbook of Ecosystems Theories and Management*, Lewis Publishers, 2000.

Koestler A.and J. Smythies, *Beyond Reductionism*, Hutchinson, 1969.

Langton C., « Studying Artificial Life with cellular automata », *Physica D*, vol. 22, 1986.

Le Moigne J.L. *Théorie du système général* PUF Paris 1994

Leonard A., « Vortex methods for flow simulation », *Journal of Computational Physics*, vol. 37, 1980, p. 289-335.

Lesieur M., *Turbulence in fluids*, Martinus Nijhoff Publishers, 1987.

Marcenac P., « Modélisation de Systèmes Complexes par Agents », *Techniques et Sciences Informatiques*, , 1997.

Odum E. P. and H. T, *Fundamentals of ecology*, Philadelphia, W. B. Saunders Company, 1953.

Palmer J., « Hierarchical and concurrent individual-based modelling », DEANGELIS D., GROSS L., Eds., *Individual-based models and approaches in ecology*, Chapman et Hall, 1992, p. 36-66.

Servat D., E. Perrier., J.-P.Treuil, and A.Drogoul, « When Agents Emerge from Agents : Introducing Multi-scale Viewpoints in Multi-agent Simulations », *MABS*, 1998, p. 183-198.

Tranouez P., C.Bertelle and D.Olivier, « Changing the level of description of a fluid flow in a agent-based simulation », *ESS 2001 Conference*, Marseilles (France), October 2001.

Von bertalanffy L., *Genaral System Theory,* Georges Brazille inc., New York, 1968.

Von Neumann J.and A. Burks., Eds., *Theory of Self-Reproduction Automata*, University of Illinois Press, 1966.

HYBRID AND HIERARCHIC COMPARTMENTAL APPROACH FOR ECOSYSTEM APPLIED TO ESTUARIES MODELISATION

Guillaume Prévost, Pierrick Tranouez, Cyrille Bertelle and Damien Olivier
LIH - Université du Havre
25 rue Ph. Lebon - BP 540
76058 Le Havre Cedex - France
E-mail : Guillaume.Prevost@univ-lehavre.fr

KEYWORDS

Estuary, Aquatic Ecosystem, Food-Chain, Individual-Based Model, Agent-Based Simulation.

ABSTRACT

This paper deals with aquatic ecosystem model in estuaries context. These systems are described with functional organizations which are producers, consumers and decomposers. We use a hierarchical representation which is based on compartmental model. The different compartments exchange various fluxes of their global components, like oxygen for example, as well as fluxes of population individuals, like fishes for example. We propose an hybrid approach based on a mixed formulation, equation based model for global components of compartments and individual-based models for living entities. We show experiments developed in an agent-based simulation, using the Madkit platform (Gutknecht and Ferber, 1997).

INTRODUCTION

Estuaries are both the site of intensive human activities and biological phenomena. They are the site of ecosystems constituted with multiple components undergoing significant gradients (salinity, temperature, ...) (Guézennec et al., 1999). Within a harbor development project context, we need explicative and predictive models able to provide informations for decision support systems. Ecosystems models are usually based on differential equations and sometimes on individual-based approaches. Our goal is to provide a general model making mix of these two approaches easier. Our modelization is based on the fact that ecosystems are hierarchic complex systems, able to handle multiscale phenomena (Tranouez et al., 2001).

We present in section 2, some general considerations about ecosystem modelling in complex systems context and specific problematics in the Seine estuary context. Section 3 presents an overview of different approaches for aquatic modelling, some based on differential equations formulation and others using individual-based models. Then we describe aquatic ecosystem in term of functional organizations decomposed in producers, consumers and decomposers. Section 4 describes our modelization based on hierarchical compartmental approach. Section 5 presents experimentations using an agent-based simulation. Section 6 draws up conclusions and perspectives.

ESTUARIES ECOSYSTEMS MODELISATION

Estuaries ecosystems

Estuaries are home of particular ecosystems. The understanding of these ecosystems is important as they own a crucial place for human civilization (industry, trade, fishing, ...). Thus, they are exposed to human influence (contamination, buildingss ...) (Guézennec et al., 1999).

Estuaries are also important biological places as hatchery for juveniles or rest-places for migrating species. In addition, they are the site of many different environmental conditions and phenomena, so estuaries are complex systems meaning that they can't be simplified without a drastic loss of information (Le Moigne, 1994). Furthermore, they are open systems as they are continuously crossed by energey fluxes that structure them (Frontier and Pichot-Viale, 1998). Organizations as sub-systems appear and evolve. They lead to hierarchical systems containing many levels of description. In such systems, the different components interact inside a given level but there are also multi-scale interactions. Furthermore, energey fluxes specific to open systems, can be considered as interactions between two scale levels.

Ecosystems have a life cycle. First, they produce lot of materials and then they make complex by increasing its biodiversity. Then, their complexity is lowering to provide the ecosystem an economical functioning. Subsequently, natural events called stresses are able to break complex system equilibrium. They lead an ecosystem to decrease its complexity level to an anterior state. Fluxes between different organizations of hierarchic systems can produce stress or, at the opposite, increase its complexity.

The Seine estuary

The Seine estuary, which is our main concern, demonstrates the multiple aspects of those ecosystem. That estuary is exposed to two daily tides. The tides are confronted to the flow from La Seine. Those flows bring about material transport, erosion, settlement, mud formation ... The importance of such phenomena is proportional to the power of tides. Ther are studies dealing with the behaviour, formation and displacement of the maximum turbidity zone. That zone is typically an emergent complex system composed of various organizations (chemical, living individuals, ...) evolving under the influence of many fluxes. Thus, the Seine estuary is a multi-scale ecosystem where hierarchic systems are relevant. Some of its components can be represented by global description, where equational systems are relevant, and others by behaviour based description, where individual-based models are relevant. So a hybrid model will be used.

AQUATIC ECOSYSTEMS MODELING OVERVIEW

Differential Approach Versus Individual-Based Approach

In ecology, many models are based on differential equations which model population dynamics using global descriptors (for example, average size, number of individuals, ...). These classes of models are not well-suited to represent on one hand, some specific individual behavior in populations and on the other hand, spatial influence of local phenomena over the whole system. Ecology shows the fundamental effects of these two aspects in the development of structural organizations in ecosystem. Subsequently, populations and ecosystems themselves influence the individuals states. There was a blatant need to represent individual as they are the link between local or spacial events and the population. The arrival of Individual-Based Models (IBM) in Ecology induces an evolution of differential models. As proposed by Haefner (Haefner, 1992), we can consider three main classes of models:

- *p-state* (population state) where population are globally represented as a linear combination of its own individuals;
- *i-state distribution* models where populations are structured in sub-classes, taking into account some specific characteristics like age, height, fertility, ... Inside these sub-classes, the individual variability is not considered (mixing hypothesis);
- *i-state configuration* models where each individual is represented as a specific entity.

Computer sciences applied the concept of agent or object to the IBM's, achieving simulations driven by the entities themselves. The consideration of the retroaction of local behaviour on organization found a new impulse here. In this case, populations as emergent organizations are described by a non linear combination of their individuals.

Aquatic Ecosystem Description Based on Functional Organizations

Aquatic ecosystems continuously produce many types of substances, such as living organisms and mineral material. A cyclic phenomenon of transformation is developed, showing a three parts system :

- *Producers* are essentially vegetable constituents which generate organic material based on light radiation transformation (photosynthesis);
- *Consumers* are composed of many types of living animals which consume the previous producers or other types of consumers in a classical food-chain process. Standard differential models used to describe their behaviour are preys-predators (Lokta-Volterra) equations.
- *Decomposers* are micro-organisms, like bacteria, which consume inert organic material such as dead animal or various secretions. Their role is to transform such complex material in mineral substance which is needed by the producers.

The model proposed in the following is based on a hierarchical approach containing organizations. Each of these organizations is composed of simple ecosystem descriptions based on the previous three fundamental parts description.

HYBRID MODEL FOR HIERARCHICAL COMPARTMENTAL ECOSYSTEMS

We previously described several aspects of modelization in ecology. Basing on these concepts, we propose an hybrid model for ecosystems simulations. This model represents both global characteristics like oxygen (O2) concentration and individual behaviours of fishes for example.

The basic level is an individual based one. We assume that individuals constitute the lower level of an ecosystem functioning. Moreover, we represent a structured system decomposed in several compartments which exchange various fluxes. Each compartment contains functional organizations such as food-chains associated to mineral components able to describe a three parts ecosystem based on producers, consumers and decomposers. Each compartment represents a functional organization like individuals aggregation. Such an individual aggregation is considered as an upper-level in the hierarchical representation. It can be considered as an *i-state distribution* model as previously defined. So, we define the first step of a multiscale representation, each compartment can be recursively aggregated with other ones.

In figure 1, we sketch such a global hierarchic compartmental system. This one contains two fish species population and a phytoplankton population. One fish population is a predator for the two other populations. The other fish population only feeds on plankton. The plankton is responsible for the photosynthesis and produces oxygen. The fishes need oxygen and

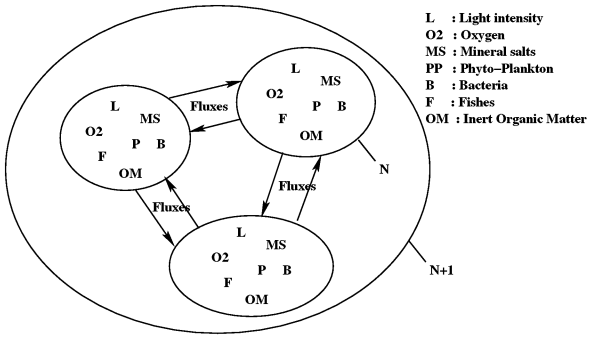


Figure 1: Global Hierarchic Compartmental System

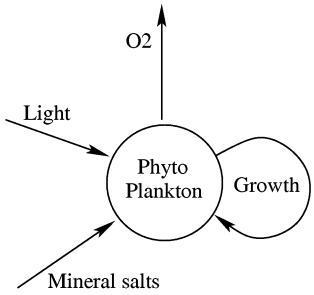


Figure 2: Producer - phytoplankton

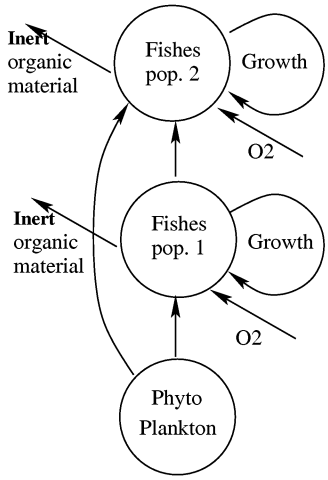


Figure 3: Consumer - trophic chain

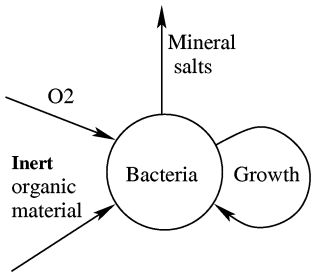


Figure 4: Decomposer - bacteria

food to survive. This theoretical representation introduces a critical resource (the phytoplankton). These trophic chains are placed in several ecosystems that are linked by individual exchange (migration) and O2 flows.This example based on a typical trophic chain introduces consumers, producers and decomposers which functional description is summed up in the three figures 2, 3 and 4.

AGENT-BASED SIMULATION

We have used an agent-based framework, called Madkit (Gutknecht and Ferber, 1997), to realize our first simulations (Cognard and Perrier, 2003). Ecosystems have their own explicit space modelled as a two-dimensional grid. Each compartment defines global states for luminosity and oxygen. Three populations are living within those compartiments. Plankton is not moving and is responsible for the photosynthesis depending on the luminosity variable. The oxygen produced is added to a global concentration managed for each compartment. Plankton is multiplying using an energy state evolving with photosynthesis. Fishes are moving in the space and are elements of a prey-predator process. The reproduction is managed with an energy state evolving with prey consumption. If they starve, their energy state gets lower. When it becomes nil, they die. Compartments are exchanging fluxes like oxygen and fishes.

We show in figure 5, the desktop managing the simulation. Some of the inner windows show a spatial distribution of all the populations of a specific compartment and others, those associated to the last ones, show the evolution curves of the number of individuals of each these populations.

The results obtained with this experimental framework lead sometimes to the constitution of compartments which concentrate a specific population. In fact, one compartment becomes a plankton pool and so there is an oxygen producer which transfers fluxes to other compartments where for example, some fish species are dominant. So, this model is able to constitute sub-systems which globally act on their own constituents, as usually described in complex systems behaviours.

CONCLUSION AND PERSPECTIVE

Some aspects of complex organizations in ecosystem, like in the Seine estuary, can be modelised according to the approach previously presented. In (Guézennec et al., 1999), this estuary is presented as a decomposition based on some functional compartments which exchange fluxes and thus which can be expressed as sub-systems fitting our model. The next step of our work is thus to confront in-situ experimental behaviours with some computer based behaviours obtained with the proposed simulation. Using more complex descriptions for the biodiversity, we will be able to produce some explicative and predictive modelizations for auto-organization phenomena in biological systems.

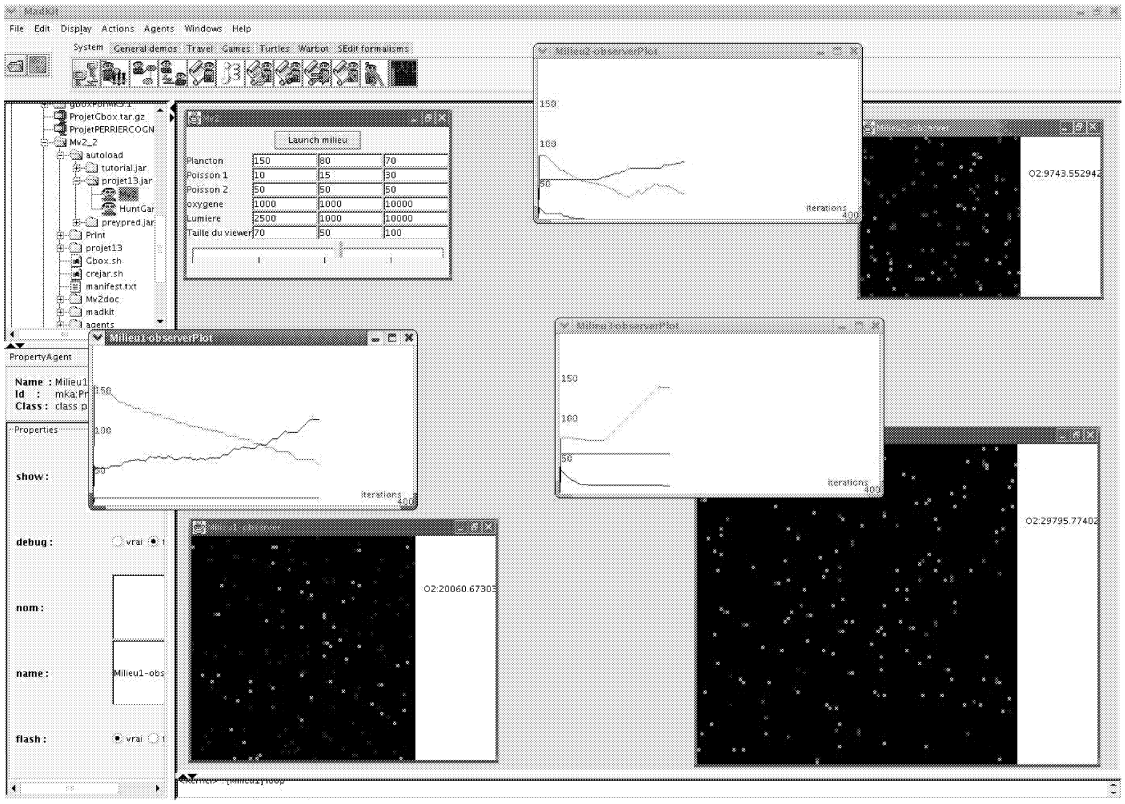


Figure 5: Agent-based simulation

GRANTS

G. Prevost thanks the "Région Haute-Normandie" for its support concerning his contribution to this study.

References

Bertelle, C., Jay, V., Lerebourg, S., Olivier, D., and Tranouez, P. (2002). Dynamic clustering for auto-organized structures in complex fluid flows. In *ESS 2002 Conf.*, Dresden (Germany).

Cognard, S. and Perrier, G. (2003). Ecosystèmes aquatiques hiérarchisés. Technical report, LIH - Le Havre University.

Coquillard, P. and Hill, D. (1997). *Modélisation et simulation d'écosystèmes*. Masson.

DeAngelis, D. L. and Gross, L. J., editors (1992a). *Individual-based models and approaches in ecology*. Chapman Hall.

DeAngelis, D. L. and Gross, L. J. (1992b). Which individual-based approach is most appropriate for a given problem. In DeAngelis, D. L. and Gross, L. J., editors, *Individual-based models and approaches in ecology*, pages 67–87. Chapman Hall.

Frontier, S. and Pichot-Viale, D. (1998). *Ecosystème*. Dunod.

Grim, V. (1999). Ten years of individual-based modelling in ecology: what have we learned and what could we learn in the furture. *Ecological Modelling*, (115):129–148.

Guézennec, L., Romana, L., Goujon, R., and Meyer, R. (1999). *Seine-Aval : un estuaire et ses problèmes*. Programme scientifique Seine-Aval. Ifremer editions.

Gutknecht, O. and Ferber, J. (1997). Madkit: Organizing heterogeneity with groups in a platform for multiple multi-agents systems. Technical report, LIRMM, Montpellier University, http://www.madkit.org.

Haefner, J. W. (1992). Parallel computers and individual models: an overview. In DeAngelis, D. and Gross, L., editors, *Individual-based models and approaches in ecology*, pages 126–164. Chapman Hall.

Le Moigne, J.-L. (1994). *Modélisation des systèmes complexes*. Dunod.

Tranouez, P., Bertelle, C., and Olivier, D. (2001). Changing the level of description of a fluid flow in a agent-based simulation. In *ESS 2001 Conference*, Marseilles (France).

THE POSSIBILITY OF QUALITATIVE MODELING OF RENEWABLE FISH RESOURCES

Ante Munitic, Merica Sliskovic, Josko Dvornik and Gorana Jelic
University of Split
Maritime Faculty of Split
Zrinsko-Frankopanska 38
21000 Split, Croatia
e-mail: amunitic@pfst.hr

KEYWORDS
System Dynamics, qualitative modelling, renewable resources, fish population system

ABSTRACT

The problem in management of the renewable resources commonly arises from the competition between resources users, what at the end causes overexploitation of the resources (tragedy of the commons). Daily it is possible to encounter some examples of inappropriate, i.e. overexploitation of natural renewable resources (such are fishes, whales, forests, terrestrial plant and animal population, even and humans). Renewable resources, in this paper fish population, typically include fish stock and natural nonlinearities which is product of complex of behaviour dynamics of the fish population system.
The main purpose of this paper is to present possibility of System Dynamics in education, research and designing in management of the fishing in some restricted area. The aim of this paper is to present the practical application of the method of the System Dynamics for qualitative and quantitative modelling of behaviour dynamics of renewable resources, i.e. fish population. Also, all cause – consequence relations between relevant variables of fish population system will be pointed out, including the dominant feedback loops. Fish population system shall be presented with mental-verbal and structural models in changeable condition of the exploitation (i.e. their ecological environments). Furthermore, the adequate model for management of fish population with optimal profits and satisfaction with all possible limitation of the environments will be given. In given model of management the main concern is not to over fish, and cause of that we will determinate optimal fishing capacities.

INTRODUCTION:

Dynamic feedback system perspective is one perspective for computer simulation modelling. Various systems can be defined as a collection of interacting elements that function together for some purpose. System thinking is concerned with connectedness and wholeness i.e. it study connection between different parts which forms a whole. Biologist Ludwig von Bertalanffy (1930) which was one of the pioneers in system thinking, observed that biological organisms functioning as a system. Many historically important simulation models are used in this paper (Gause conclusions, Lotka –Volterra models and Ricker models). The System Dynamics Modelling is in essence special, i.e. "holistic" approach to the simulation of the dynamics behaviour of natural systems, and it contains quantitative and qualitative Simulation Modelling of various natured realities.

SYSTEM DYNAMICS QUALITATIVE MODEL OF RENEWABLE FISH RESOURCES

System Dynamics mental verbal model of sub system of fish population

Short mental-verbal model of sub system of fish population

FBL1(+): FHR =>(+)FISH => (+)FHR
FBL2(-): FISH =>(+)FDR => (-)FISH
FBL3(-): FISH =>(+)DF => (+)FDR => (-)FISH
FBL4(-): FISH => (+)DEN => (+)CPS => (+)TCPY => (-)FISH

"When variable hatch fraction (HF) is increasing fish hatch rate (FHR) shall be increased too, resulting in positive (+) dynamics character of CCL (cause-consequences link)."
"When variable fish hatch rate (FHR) is increasing fish population (FISH) shall be increased too, resulting in positive (+) dynamics character of CCL (cause-consequences link)."
"When variable fish population (FISH) is increasing fish hatch rate (FHR) shall be increased too, resulting in positive (+) dynamics character of CCL (cause-consequences link)."

"When variable fish population (FISH) is increasing fish death rate (FDR) shall be increased to, resulting in positive (+) dynamics character of CCL (cause-consequences link)."
"When variable fish death rate (FDR) is increasing consequently fish population (FISH) shall be decreased, resulting in negative (-) dynamics character of CCL (cause-consequences link)."

"When variable fish population (FISH) is increasing death fraction (DF) shall be increased too, resulting in positive (+) dynamics character of CCL (cause-consequences link)."

"When variable carrying capacity (CC) is increasing death fraction (DF) shall be decreased, resulting in negative (-) dynamics character of CCL (cause-consequences link)."
"When variable death fraction (FHR) is increasing fish death rate (FDR) shall be increased too, resulting in positive (+) dynamics character of CCL (cause-consequences link)."
"When variable fish death rate (FDR) is increasing consequently fish population (FISH) shall be decreased, resulting in negative (-) dynamics character of CCL (cause-consequences link)."

"When variable fish population (FISH) is increasing, density (DEN) is also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."
"When variable density (DEN) is increasing, catch per ship (CPS) is also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."
"When variable area (AREA) is increasing variable density (DEN) shall be decreased, resulting in negative (-) dynamics character of CCL (cause-consequences link)."
"When variable catch per ship (CPS) is increasing, total catch per year (TCPY) is also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."
"When variable total catch per year (TCPY) is increasing, fish population (FISH) is decreasing, what results in negative (-) dynamics character of CCL (cause-consequences link)."
"When a variable ship (SHIP) is increasing, total catch per year (TCPY) is also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."

System Dynamics mental verbal model of sub system of the ships

Short mental-verbal model of sub system of the ships

FBL5(-): SHIPS => (-)YEARP => (+)SBR => (+)SHIP
FBL6(+): SHIPS => (+)TCPY => (+) REVEN => (+) YEARP => (+) SBR => (+) SHIPS

"When variable ships (SHIP) is increasing, yearly profits (YEARP) is decreasing, what results in negative (-) dynamics character of CCL (cause-consequences link)."
"When a variable yearly profit (YEARP) is increasing, ship building rate (SBR) is also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."
"When a variable ship building rate (SBR) is increasing, number of ships (SHIP) is also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."
"When variable ship costs (SHIPC) is increasing, ship building rate (SBR) is decreasing, what results in negative (-) dynamics character of CCL (cause-consequences link)."
"When a variable fraction invested is increasing, ship building rate (SBR) is also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."
"When variable ships (SHIP) are increasing, total catch per year (TCPY) is also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."
"When variable total catch per year (TCPY) is increasing, a revenue (REVEN) is also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."
"When variable catch per ship (CPS) is increasing, total catch per year (TCPY) is also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."
"When variable revenue (REVEN) is increasing, a yearly profit (YEARP) is also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."
"When variable fish price (FISHP) is increasing, variable revenue (REVEN) is also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."
"When variable yearly profit (YEARP) is increasing, a ship building rate (SBR) is also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."
"When variable operating costs per ship (OCPS) is increasing, a yearly profit (YEARP) is also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."
"When a variable ship building rate (SBR) is increasing, number of ships (SHIP) is also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."

System Dynamics mental verbal model of sub system of profit

"When variable yearly profit (YEARP) is increasing, profits per year (YEARP) is also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."
"When a variable profit per year (YEARP) is increasing, a total profit (TOTPR) is also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."
"When a variable total profit (TOTPR) is increasing, total assets (TOTASS) are also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link)."
"When a variable number of ships (SHIPS) are increasing, total assets (TOTASS) are also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link).
"When a variable salvage value per ship (SVPS) is increasing, total assets (TOTASS) are also increasing, what results in positive (+) dynamics character of CCL (cause-consequences link):"

STRUCTURAL SIMULATION MODEL OF RENEWABLE FISH RESOURCES

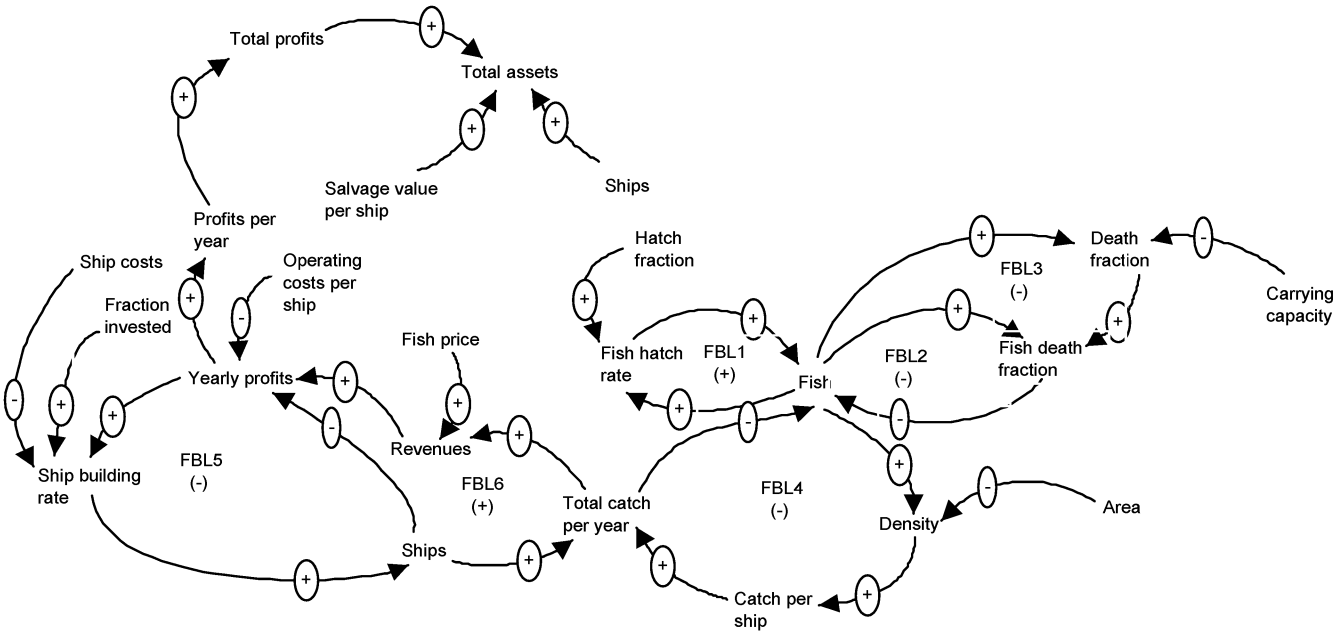


Figure 1. Structural simulation model of renewable fish resources

CONCLUSION

Various systems can be defined as a collection of interacting elements that function together for some purpose. In this paper we tried to point out important elements of the fish renewable resources system and to present their mutual relationships (cause – consequence links). Qualitative modelling i.e. mental - verbal model gives the possibility to notice each element of the system separately and to describe its cause - consequence link with others elements of the system. Observed system of fish renewable resources is divided in three subsystems (subsystem of fish population, subsystem of ships and subsystem of profits). The connection between these subsystems is evident from structural diagram including the dominant feedback loops. System thinking is concerned with connectedness and wholeness, the both conditions that are satisfied in this paper. System Dynamics qualitative models of renewable fish resources, as distinguished from historical models, gives an additional possibility to observe dynamics character of behaviour of the cause consequence variables, also with a possibility to obviously distinguish which variable is cause and which is consequence. In FBL cause variable can be cause which effects other variable, and in the same time it can be consequence which are affected by second or third variable. This possibility to observe the dynamics of behaviour of models variables is very suitable for qualitative investigation of dynamics behaviour of fish resources.

REFERENCES

Constanza, R., Ruth, M. 1998." Using Dynamics modelling to Scope Environmental problems and Build Consensus." Environmental Management Vol.22, No.2, pp 183-195.

Cushman-Roisin, B., Rice III, NJ., Moldaver, MA. 1999. *A Simulation Tool for Industrial Ecology: Creating a Board Game.* Journal of Industral Ecology, Vol.3, No.4, pp 131-144.

Forrester, JW. 1961. *Industrial Dynamics.* MIT Press: Cambridge, MA.

Forrester, JW. 1968. *Principles of Systems.* Wright-Allen Press: Cambridge, MA.

Forrester, JW. 1971. *World Dynamics.* Wright-Allen Press: Cambridge, MA

Hastings, A. 1997. *Population biology, Concepts and Models.* Springer- Verlag, New York, Inc.

Lindholm, J.B., Auster, PJ., Ruth, M., Kaufman, L. 2001. *Modelling the Effects of Fishing and Implications fort he Design of Marine Protected areas: Juvenile Fish Responses to Variations in seafloor Habitat.* Conservation Biology, Vol.15, No.2, pp 424-437.

Munitic, A. *Computer simulation with the help of System Dynamics*, in Croatian, Brodosplit, 1989, Split, Croatia (299 pages)

Meadows, DL. *Fish bank website.* Available[online] 22.08.2002. http://www.unh.edu/ipssr/Lab/FishBank.html

Moxnes, E. 2000. *Not only the tragedy of the commons: misperceptions of feedbacks and policies for sustainable development.* System Dynamic Review, Vol. 16, No.4, pp 325-348.

Roberts, N., Andersen, D., Deal, R., Garet, M., Shaffer, W. 1983. *Introduction to computer simulation: The System Dynamics approach.* Addison-Wesley Publishing Company, Inc., MA.

Richardson, G.P. 1991. *Feedback Thought in Social Science and System Theory.* University of Pennsylvania Press.

Schaefer, MB. (1957). *Some consideration of population dynamics and economics in relation to the management of marine fisheries.* Journal of the Fisheries Research Board of Canada, Vol. 14, pp 669-681.

Whelan, J., Halbower, M. *Building the Fish Banks Model & Renewable Resources Depletion (D-4543).* Available [online] from Roadmaps, http://sysdyn.mit.edu/road-maps/rm-toc.html

Agent Modeling of the Caparo Forest Reserve

Magdiel Ablan,
Jacinto Dávila,
Niandry Moreno,
Raquel Quintero,
Mayerlin Uzcátegui.
Centro de Simulación y Modelos (CESIMO).
Universidad de los Andes. Facultad de Ingeniería.
Av. Don Tulio Febres Cordero. Mérida, Venezuela.5101
mablan@ula.ve, jacinto@ula.ve, morenos@ula.ve, raquelq@ula.ve, maye@ula.ve

KEYWORDS
Environmental Science, Simulators, AI in simulation, AI-supported simulation, Object Oriented.

ABSTRACT

This article presents a preliminary model of the biocomplexity of a Venezuelan forest reserve. The computational model describes the interactions between human systems and their environment, following the guidelines of a multi-agent simulation theory. Land use changes are simulated using Cellular Automata. This project is supported by a multi-agent simulation platform called GALATEA, and the Cellular Automata library of SpaSim.

1.- INTRODUCTION

This study is a subproject of Biocomplexity: Integrating Models of Natural and Human Dynamics in Forest Landscapes Across Scales and Cultures (http://www.geog.unt.edu/biocomplexity). It aims to model and simulate land use processes and changes in vegetation cover as a consequence of human actions.

The National Science Foundation (NSF) defines Biocomplexity as the resultant phenomenon of dynamic interactions among biological, physical, and social components at different environmental systems of the earth (North Texas University, 2003). The purpose of these studies is to achieve an understanding of those interactions and to be able to analyze and possibly forecast their effects.

As a first approximation, we have devised a set of toy models to cater for 1) the human dynamics, using a set of conceptual tools and data structures provided by GALATEA (Uzcátegui, 2002) and 2) the environmental dynamics, by integrating a cellular automaton from the SpaSim (Moreno, 2001, 2002) library into the actual simulator of the reserve. The data structures of Galatea provide for the representation of the agents' goals, beliefs and observations, and, also, for a very elementary reasoning engine to deduce actions for each agent, according to its circumstances.

We started by developing an elementary model ("toy model #1") that simulates some of the interactions that occurs at Caparo Forest Reserve. As we improve our platform capacity we will include new features to our toy models aiming to achieve real system complexity.

2.- CASE STUDY

Various processes have attempted against the sustainability of one of the oldest forest reserves of Venezuela, "the Caparo Forest Reserve (CFR)". As other reserves, this was created to guarantee the maintenance of the logging industry in the zone, and, at the same time, to preserve one of the best forests of Venezuela. Regrettably, these good purposes were never achieved, and in a few decades the reserve went from a forest, that practically had not suffered human intervention, to a highly deforested zone, occupied by many settlers (Tonella et al., 1998).

CFR was created in 1961 and it's located at Southeast of Barinas State, in the Venezuelan western plains region. It had 176.434 hectares, divided on three units to facilitate its management. Our study takes place at Unit I, which had 53.358 hectares, and includes a special area called the Experimental Unit, that was assigned to Los Andes University to develop research and educational activities.

Many factors have contributed to forest disappearing at RFC, such as unsuitable forest management of some lumber concessionaires (Government had assigned Forest Reserve Units to some Logging Companies in concession form), contradictions between different governmental organisms, poverty and the demand of lands for agricultural activities, and the existence of political interests in favor of settlements, among others factors. All of them have fomented the establishment of settlers, and consequently, the advance of the farming activities in wooded zones without any control.

Currently, just 7.000 hectares of forest at RFC have survived, which are located at the Experimental Unit and are not exempted of future deforestations due to agrarian settlement process such as the ones previously described.

According to Rojas (Rojas, 1993), the agrarian settlement process of forest reserves at the Venezuelan western plains has developed through the following stages:

- Primary Cycle: The first settlers take possession of a certain area at the reserve and practice subsistence (i.e. slash and burn) agriculture. This surface can be an uncultivated land (previously deforested and unoccupied) or can be forest land deforested by the settler. Before five years, the soils are exhausted, and the harvests are no longer enough to sustain the settler and his family. Some settlers try by expanding their farms at the expense of new deforestation. However, sooner or later, they will end facing the same situation. The alternative is to seed pasture (which gives value to the land) and later, to sell its improvements to landlords or other settlers, luckier than them, whom soon will be using the initial settlers as their workers.

- Land Market: Land improvements are sold and they are registered. Then settlers can buy off new land to other more recent settlers, or just return to initiate a new primary cycle of invasions. If there is some money left, they would generally prefer to invest in cattle because it is much more profitable and it does not have as many disadvantages as agriculture. But definitively, the settler returns to invade because he has the hope to receive money again if things do not go well for him (Tonella et al., 1998). At this stage, the pasture retailers and landlords acquire the improvements of primary settlers. Extensive cattle ranch dominates the land use. After years, the property of the parcels is transferred to the settlers, by application of the Agrarian Reform. Then they are sold at ridiculous prices, to the landlords, politicians and cattle dealers who urged and supported the original settlements (Centeno, 1997).

- Cattle Ranch Consolidation: Cattle ranching is the main activity. Landlords use previous deforestation done by the settlers, buy their improvements, and acquire greater extensions for cattle raising, (this activity is developed as a capitalist company by landlords). This process, characterized by the concentration of the property, forces the initial group to move towards primary cycle settlements or to wage-earning work (Sánchez, 1989).

3.- THE MODEL

During this first stage of the project we developed a simplified model that considers several instances of settler agents and a lumber "concessionary" agent, whose behaviors will be described next.

3.1.-Settler Agent Characteristics

The settlers are people of limited economical resources that arrive at the reserve aiming to improve their economical status and to obtain the property of the land that they occupy. Initially they are dedicated to subsistence agriculture and they just try to maximize the benefits from their occupation in the area, without considering soil exhaustion due to inexistent or poor management practices, and without taking into account the ecological damage that might be caused. After five years, the land loses its fertility, and the settler must move to another place or expand his farm by deforesting some adjacent land.

The behavior of the settler agent is defined by all the actions that it executes within the reserve from the initial moment of occupation or invasion to the moment of his abandonment of the reserve. The following items describe his behavior:

1.- When a settler arrives at the reserve, he observes the environment around him.

2.-He looks for a unoccupied place without vigilance to settle down.

3.- Once he has settled down at the forest reserve, he proceeds to deforest, to clear the land of weeds or to harvest some crop for his subsistence, depending on land use. In the clearing process trees are destroyed and the settler could obtain some monetary income from the sale of any existing wood of commercial value.

4.- Settler sells wood of commercial value. In this first model, this will be the only way to receive a monetary income, because the commercial transactions between settlers due to land improvements are not considered.

Table Number 1 shows detailed settler agent characteristic:

3.2.-Rules for Settler Agent

To detail settler agents rules we use Actilog Language, which is a language to write generalized, condition --> action, activation rules. The semantics of the language is based on the assumption that implications (conditional goals) can be used to state integrity constraints for an agent. These integrity constraints describe conditions under which the agent's goals must be reduced to plans that can be executed. For instance, a rule such as if A then B, will indicate to the agent that whenever it can prove that A is the case, it then should pursue goal B. B is normally the description of a task that must be reduced to a set of low-level, primitive actions that the agent can execute. See (Dávila, 2003) for more details.

For settler agent we have several rules:

- ZONE VIGILANCE: A vigilance zone map will be considered, including neighborhoods to the National Guard control places, the organism in charge of the vigilance of the reserve, and the effect of the forest plantations monitoring (plantations of less than two years) by the logging concessionaires. The following code line in Actilog language establishes the rule that will govern the behavior of settlers in case of existing vigilance at the zone.

Table 1: Settler Agent Characteristics

Percepts	Internal State	Environment	Possible Decisions	Actions	Goals
To observe the environment (zone vigilance, land use, land occupation)	-Economical Status. -Land Ownership	-Level of zone vigilance. -Land fertility. -Land use (uncultivated and unoccupied, plantations, forest with or without commercial valuable species, agriculture, farming or cattle activities). -Land occupation.	-To harvest crops for own subsistence -Moving -Deforesting -Invading unoccupied lands. -Expand farms. -Illegal selling of wood.	-To harvest crops for own subsistence. - Farming activities. -Moving -Deforesting -Invading unoccupied lands. -Expand farms. -Illegal selling of wood.	-To improve economical status. -To become a landlord. -To get legally ownership of the land at the reserve.

```
if vigilancia then pensar_en_otro_sitio .
```

- SETTLER INITIAL ESTABLISMENT: If there is not vigilance at the zone and zone is unoccupied, the colonist can settle down in the chosen area. Vegetation is destroyed, high commercial valuables woods are sold and farms or harvests are established. The next Actilog language code line indicates establishment rules for the settlers in the reserve.

```
if vigilancia then pensar_en_otro_sitio.
if not(tierra_ocupada), not(vigilancia),
uso_de_la_tierra_igual_baldia then establecerse1.
if not(tierra_ocupada), not(vigilancia),
uso_de_la_tierra_igual_bosque_sin_madera then
establecerse2.
if not(tierra_ocupada), not(vigilancia),
uso_de_la_tierra_igual_plantacion then establecerse2.
if not(tierra_ocupada), not(vigilancia),
uso_de_la_tierra_igual_bosque_con_madera then
establecerse3.
to establecerse1 do invadir,
to establecerse2 do invadir, talar,
to establecerse3 do invadir, talar, venta_de_madera.
```

- FARMS EXPANSION: There is only one possibility for farm expansion of fundos, which is carried out at neighboring unoccupied land, because commercial transactions between settlers are not considered. The next Actilog language code line indicates the only way to farm expansion:

```
if pensando_en_sitio_vecino, not(tierra_ocupada),
not(vigilancia) then expandir_fundo.
```

3.3.- Concessionary Agent Characteristic

The lumber concessionaires are represented by private companies that have the function to carry out the forest exploitation and management plans in the reserve areas under the supervision of the Ministry of Environment. These companies must develop the necessary infrastructure for the logging industry in the zone and protect their assigned area at the forest reserve (Sanchez, 1989).

At this stage, the lumber concessionary agent implemented, makes a very simplistic and hypothetical forest management within the reserve: the lumber concessionaire exploits the forest and proceeds to plant commercial valuable species regardless of the native species or the initial physical and ecological conditions in the area; furthermore, the concessionaire is in charge of forest plantations vigilance during the first two years. If concessionaire finds a colonist on its assigned zone, he will ignore the settler and continues the work at another place that is not occupied by settlers.

Table Number 2 shows detailed concessionary agent characteristic:

3.4.- Rules for Concessionary Agent

It is assumed that there is just one concessionary agent operating at our forest reserve area. Its behavior is determined by following rules:

- If the concessionaire finds a forest area that is not occupied by settlers, will proceed to explode it and plant the place with high commercial valuable species.
- If the concessionaire finds a place uncultivated and unoccupied proceeds to establish a forest plantation on it.
- It is important to notice that concessionaire will watch this plantation during the first two years, according to what was established by the Government at the Concession contract.

Next Actilog language code line shows concessionaire behavior:

```
if uso_de_la_tierra_igual_bosque_sin_madera, not
  (tierra_ocupada_por_colonos) then explotar.
if uso_de_la_tierra_igual_bosque_con_madera, not
  (tierra_ocupada_por_colonos) then explotar.
to explotar do tumbar_arbol, transportar, plantar.
if uso_de_la_tierra_igual_baldia, not
(tierra_ocupada_por_colonos) then plantar.
to plantar do sembrar, establecer_vigilancia.
if plantacion_mayor_a_dos_años then quitar_vigilancia.
```

4.- IMPLEMENTATION

The simulation platform GALATEA integrates the concepts and tools that allow for the simulation of systems under the distributed, interactive, continuous, discrete and hybrid approaches. But, it also, incorporates support for modeling and simulation of multi-agents systems.

The simulation platform GALATEA consists of:

- A family of languages to model systems multi-agents,
- A simulator
- A compiler still under construction, to translate model from the higher level languages offered to the modelers, into Java, the basic language of the simulator.
- A framework for the construction of models and simulation.

GALATEA is an extension of the DEVS oriented platform GLIDER (GLIDER is a language and a platform for continuous and discrete events models of simulation that has a modeling style "oriented to the network": the components of the modeled system are organized in networks of nodes that interchange messages) (Domingo, 1998).

GALATEA differs from its predecessor in the addition of agents to the syntax and semantics of the modeling languages. This has many subtle but important implications. With agents, we have a concurrent model of computation for simulation :

1.- We need to explain what agents are, both at an abstract level (for the modeler) and at concrete level (for the simulator);
2.- We have to provide for languages and strategies to program agents to do things.

The simulation theory that explains the way we combined the simulator with the agents is presented in (Moreno, 2002). The implementation of the simplified model at this stage of the project was done in Java, using the library of components of the platform of simulation GALATEA and the SpaSim.

SpaSim is a software that allows the specification, simulation, visualization and analysis of spatial models in the same environment, using a friendly user interface while at the same time providing considerable flexibility. Square cells were used for the cellular automata to keep compatibility with most raster GIS systems in use.

Also the software integrates simulation techniques (like cellular automata), spatial analysis, spatio temporal analysis, and maps visualization.

The implemented model is a multi-agents spatial explicit model, where the agents are codified using GALATEA agents library, while the space is modeled as a cellular automata representing a simplified dynamics of the environmental system. The cellular automata is implemented by means of the SpaSim-lib library.

The dynamic of the environment is modeled as a cellular automata with the following characteristics:

- Number of layers: 4.
  1. Land Uses Layer: each cell can be in any of six states which are equivalent to the set {0,1,2,3,4,5,6} {“Uncultivated”, ”Agriculture”, ”Cattle Ranching”, ”Plantation”, ”Forest that includes high commercial value species”, “Forest without high commercial value species” ,”Roads”}.
  2. Time in Use Layer: used as a time count layer that indicates the time that a cell has spent remaining at a determined state (indicated at Land Uses Layer).
  3. Vigilance Layer: each cell can be in some of the following states: - 0, that represents a no watched over cell, - and 1, that indicates a watched cell.
  4. National Guard Control Place layer: similar to Vigilance Layer but it is static. It indicates the fixed positions of the control places or permanent monitoring zones in the reserve. Each cell can be in some of the states: 0, that represents that the cell is not under monitoring of any control post, and 1, that indicates the opposite
- Von Neumann Neighborhood (Von Neumann, 1996) for every cell.
- State Transition Rules:
  - An uncultivated land can stay at that state during 20 years, then it will become a forest without high commercial valuable species.
  - A plantation cells stays in that state for 20 years, then it will become a forest with high commercial value species.
  - During the first two years a plantation a cell is watched or guarded.
  - Permanent states are communication ways, forest with or without high commercial valuable species, agriculture activities or farms.

Figure 5 shows UML packages diagram (Muller, 1997) of the implemented model.

Caparo package implements the classes for the settler agent, called ColonoInicial, and the lumber-concessionary agent, called Concesionaria; both classes extend the Ag class in the package galatea.gloria. The Ag class defines the basic structure for any agent. Also, in the Caparo package, the

environment is modeled as the Ncaparo class, which extends the NodeCellularHistory (in the package spasim) class. NodeCellularHistory states the general structure of a cellular automaton. The actual simulation model is implemented as the DemoCaparo class.

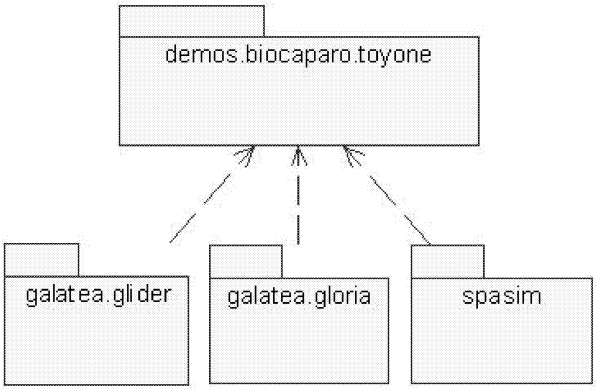


Figure 5. UML Packages Diagram of implemented model.

The model was simulated for 120 time units. Each time unit represents six months. There are just 20 settler agents and one concessionary agent considered.

For the concessionary agent there were considered an operational square regions of 6x6 cells. The Figure 7 shows Use Land Legend, that was used for Figures 8 to 10 (maps at different simulation times using the SpaSim).

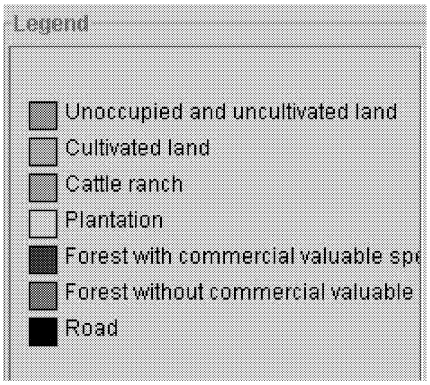
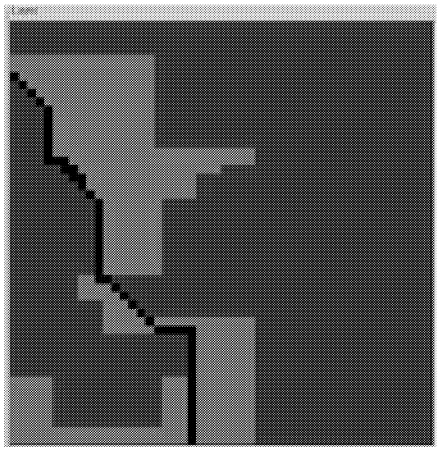
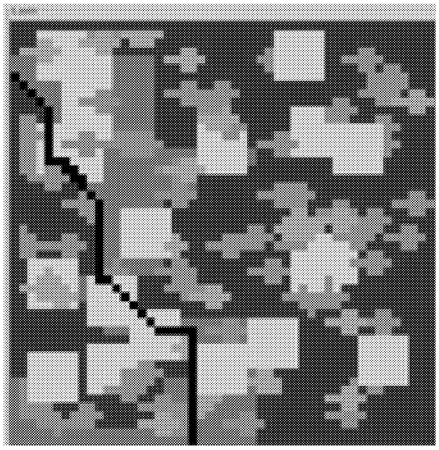
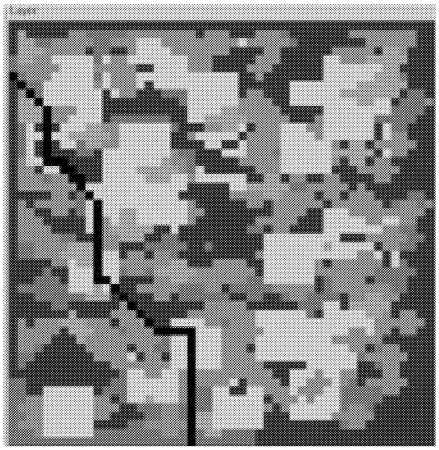


Figure 7: Land use Legend


Figure 8: Land use map for 0 simulation years


Figure 9: Land use map for 20simulation years


Figure 10: Land use map for 60 simulation years

5.- CONCLUSIONS

A working model showing the coupled dynamics of a simple environmental and social systems was presented. The model incorporates some of the basic process that cause land use change in the Caparo forest reserve and allows appreciation of the spatial patterns of deforestation over time

The Figure 11 shows how the forest areas decrease due to agents actions. Settler agent occupy forest reserve areas (to use them for agricultural activities) and, after five years, the settler agent abandons the area and moves to another one. As a result, unoccupied and uncultivated land percent increases. At the same time, concessionary agent explodes forest areas to transform them in plantations.

Through the time, the forest areas tend to disappear while unoccupied and uncultivated lands increase.

For future works we are considering the inclusion of new agents and the inclusion of more complex behavior rules for the existing ones modeled here; incorporation of real maps of land use and soils classification of the area; among others. At the same time, we will continue with the development of the GALATEA simulation platform.

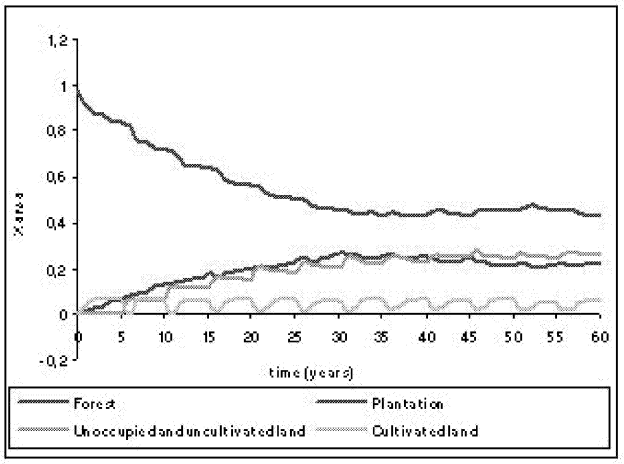


Figure 11: Land Use Distribution through 60 years.

6.- ACKNOWLEDGMENTS

Thanks to University of Los Andes' CDCHT; projects I-524-95-02-AA and I-667-99-02-B.

SpaSim software is a product of the researching activities developed by Niandry Moreno as a part of a researcher development program (Programa de Desarrollo del Investigador Nobel) at Los Andes University (ULA-FONACIT).

This material is based upon work supported by the National Science Foundation under Grant CNH BCS-0216722.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

We would like to thank Armando Torres, Hirma Ramirez, Rodrigo Barros and Emilio Vilanova for many valuable discussions and feedback.

7.- REFERENCES

Acevedo Miguel, et al. Agent Based Modeling of Human and Natural Systems and their Interactions. North Texas University. 2003.

Centeno, Julio Cesar. Deforestaciones fuera de control en Venezuela. Universidad de los Andes, 1997. http://www.ciencias.ula.ve/~jcenteno/DEFOR-ES.html]

CESIMO. Gaia. Caso Venezuela: Deforestación y Políticas de la Propiedad de la Tierra en la Reserva Forestal de Caparo. Centro de Simulación y Modelos de la Universidad de los Andes, Mérida, Venezuela. 1998. http://chue.ing.ula.ve/GAIA/CASES/VEN/ESPANOL/datos-especificos.html,

Dávila, Jacinto. Actilog: An Agent Activation Language. Practical Aspects of Declarative Languages, Dahl, V and Wadler, P. Editors. Lecture Notes in Computer Science. 2562. Springer. 2003.

Dávila, Jacinto y Tucci, Kay. Towards a logic-based, multiagent simulation theory. Conferencia Internacional en Modelado, Simulación y Redes Neuronales. MSNN 2000. Best of AMSE. Mérida. Venezuela. 2000.

Domingo, Carlos (1988). Glider, a network oriented simulation language for continuous and discrete event simulation. International Conference on Mathematical Models.

Moreno, N.; Ablan. M.; Tonella, G. '*SpaSim: A software to Simulate Cellular Automata Models*'. In proceedings IEMSs 2002, First Biennial Meeting of the International Environmental Modeling and Software Society, Lugano, Suiza. 2002.

Moreno, N.; Ablan, M. '*Librería de Componentes para desarrollo de Modelos de Simulación Espacial usando Autómatas Celulares.*' In proceedings XXVII Conferencia Latinoamericana de Informática, Mérida, Venezuela. 2001.

Moreno S., Niandry Leet. Diseño e Implementación de una estructura para el soporte de simulación espacial en Glider. Master Theses. Graduate Computer Programme Universidad de los Andes, Mérida, Venezuela, 2001

Muller, P. '*Modelado de objetos con UML*'. Eyrrolles. Ediciones Gestión 2000, S.A. Barcelona, España. 1997.

North Texas University. Proposal to the National Science Foundation (NSF): "Biocomplexity: Integrating Models of Natural and Human Dynamics in Forest Landscapes Across Scales and Cultures", 2003. http://www.geog.unt.edu/biocomplexity/

Rodríguez P., Luis E., et al. Investigación Silvicultural con fines de manejo del Bosque Tropical Alto en la Unidad Experimental de la Reserva Forestal Caparo, Venezuela. 2000.

Rojas López, José. La Colonización Agraria de las Reservas Forestales: ¿un Proceso sin Solución? Universidad de los Andes, Instituto de Geografía, Mérida, Venezuela, Cuaderno Geográficos No. 10, Junio de La Colonización Agraria de las reservas Forestales: ¿Un proceso sin solución? 1993.

Russell, Stuart J. y Norvig, Peter. Inteligencia Artificial. Un Enfoque Moderno. Primera Edición. 1996

Sánchez, M. Situación Actual del Proceso de Ocupación de la Reserva Forestal en Caparo. Universidad de los Andes, Instituto de Geografía y Conservación de Recursos Naturales, Mérida, Venezuela. 1989.

Tonella, G.; Quintero, R.; et al. 'Simulation Models, Multimedia and GIS for the Analysis of Deforestation', in Sydow, A. and Jin-Yi Yu, 'Modeling and Simulation of the Earth System', Proceedings of the 1998 Conference on Mission Earth, San Diego, 11-14 January, SCS San Diego, USA, ISBN 1-56555-139-7, 41-45. 1998.

Torres Lezama, Armando et al. Informe Final. Visión estratégica de la estación de Investigación Caparo. Facultad de Ciencias Forestales y Ambientales. Universidad de Los Andes, Mérida, Venezuela. Octubre, 2001.

Uzcátegui S., Mayerlin. Diseño de la Plataforma de Simulación de Sistemas Multi-Agentes GALATEA. Universidad de Los Andes, Mérida, Venezuela. 2002.

Von Neumann, J. '*Theory of Self-Reproducing Automata*'. University of Illinois Press, Illinois. Edited and completed by A.W. Burks. (1996).

SIMULATION IN HEALTH CARE MANAGEMENT

LOGISTICAL FLOW OPTIMISING IN MEDICAL CARE PROCESSES

Zuzana Hošáková, Zbyněk Mikša, Jana Dvořáková, Jaromír Dvořák and Jiří Urbánek
Brno University of Technology
Technická 2,
616 69 Brno, Czech Republic.
E-mail: zuzka.h@email.cz

Veronika Urbánková
Masaryk´s University Brno
Komenského nám. 2
601 00 Brno, Czech Republic.
E-mail: vera.urban@email.cz

KEY WORDS
Medical care, Logistic Crate Analysis, Object Life Cycle

ABSTRACT

Medical care is very complicated anthroposophy process chain. Here are presented the logistical management aspects of medical processes, which are focused to a Life Cycle of medical prosthetic aids (e.g. dentures, artificial limb and toggle etc.) and of medical disposable materials (e.g. dental amalgam filling, hypodermic needle etc.) further only Medical Solid Waste. Time dependent phases of these Product Life Cycles are characterised by a change of the logistical Value Added. It increases/ no-increases/ decreases respectively during medical care. The rate of its change is dependent on the time - space characteristics, a disposition, marginal conditions and the environments. Therefore, it is typical logistical task. The Change Rate of the Value Added is expressed by means a Transformation - τ. An influence of the τ to economical efficiency of medical care versus logistical processes (logistic pipeline) will be formalized via models of the "Logistic Crate Analysis" (LCA) method. Common modules of MS Windows software are used as the formalized tools of LCA. Optimisation procedure is managed by the criteria, which a survey is outlined at the paper.

INTRODUCTION

The Medical Solid Waste (MSW) is analyzed as recycling-able object of logistical chain of Object Life Cycle (OLC). The formal apparatus of the DYVELOP (Dynamical Vector Logistics of Processes) method (Mikša et al. 2003) serves for the description, analysis and evaluation of process relation to the MSW OLC. The OLC has generally four phases: Pr – Di – Cs – Re (Urbánek and Jurová 1998). The wastes are created and produced in all these four phases. But, the recycling process without redundant waste is final aim. The recycling must be ensured in defined Environment. The environmental approach (Urbánek and Rýznar 2002) includes of quite new concept of processes evaluation of medicine solid waste treatment. The recycling process efficiency criterion must be the lowest economical costs of waste treatment from the point of view Anthroposophy Sustainable Development (Urbánek and Skála 2001). The logistic costs create great part of waste treatment costs. The logistic Pipeline (Langford 1995) is than optimized passage through of PLC. The instrument for evaluation of costs structure is than special 2D graphs where first parameter is real time and second parameter is either a Value Added (VA) and/or a Transformation (τ). The 3D graphs are used for the waste treatment optimization. A logistical processing modality is added as a third parameter here. More complicated MSW logistic chains optimization acts at models of the „Logistic Crate Analysis“ (LCA) (Urbánek and Jurová 1998).

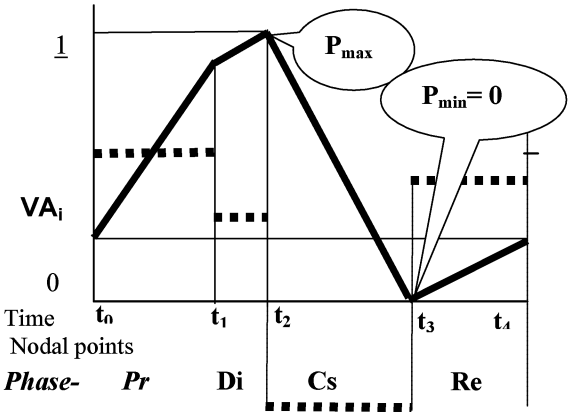


Figure 1. The value added VA_i and the transformation τ_i at OLC phases.

OLC AND TRANSFORMATION

Any Object Life Cycle (OLC) is a continuous chain of the functions (Urbánek 1999), processes and their relations at defined environment (Urbánek 2002). The OLC has four phases: Pr – production; Di – distribution; Cs – consumer; Re - recycling. The recycling includes such a processes as the re-passage, regeneration, renovation, separation, grinding, conversion, liquidation and so on. The time is a first parameter of any OLC functionality (graph) because the logistic operation flow has an expressive time - sequential character. The MSW treatment operations have continual character and they are defined as a recycling material processing where an Object is either MSW material or by patient extinguished product. The ***Transformation*** ***(τ)*** is the key time dependent functional characteristic of the Objects of medical care processes during MSW OLC. An instrument for the Transformation deduction and evaluation will be 2D graph parameterised with real time t and with Value Added VA - as an expression of processing costs (Dvořáková et al. 2002). The Transformation of recycling processed product can be defined than as a qualitative and quantitative change of technological and physics-chemical characteristics. E.g. a shape, a weight, a state, harmful properties and behaviour, a contents of harmful substance, a taste, a colour, a temperature, etc.) This change originates new function and

the new user value. The Transformation is such a characteristic of MSW OLC processes which is simply measurable and/or which is connected with the data detected from information subsystem respectively (Urbánek 2001). Common Price **P** of the OLC Object is strongly time dependent, but it is very relative at separate logistic phases of MSW OLC. The P is commonly expressed in national money units, the consequence is, that it is strongly dependent at place (space) of its occurrence. It is reason, that the Price is used only for "price margin" (**ΔP**) definition. For the simplicity, the price distribution is linear in separate logistic phases between Nodal Points at time series

$$\mathbf{t_i = \{t_0, t_1, t_2, t_3, t_4\}} \qquad (1)$$

Nevertheless, some practical *marginal condition* must be considered from sequential characters of MSW recycling processes excluding environmental damage:

a) It is necessary to secure starting price $\boldsymbol{P_0(t_0)}$ *on the higher (or equal) value then the finished price* $\boldsymbol{P_4(t_4)}$ *from the point of view of MSW recycling in identical PLC. If it is not true, the financial doping must be charged either from last patient (consumer) of extinguished object or from another source (a producer, a government, etc.).*

b) The price $\boldsymbol{P_3(t_3)}$ *must be considered without negative value to ensure of MSW OLC continuity at the Cs – phase.*

c) If the recycling (re-passage, regeneration, renovation...) operations are substituted with another type of the operations at the Re-phase (for example separation, burning, conversion, liquidation etc.) than new functional and value characteristics of the products are obtained. The primary $\boldsymbol{P_4}$ *comply with price* $\boldsymbol{P_0'(t_0')}$ *of secondary object chain must be considered (financial yielding principle).*

d) The price $\boldsymbol{P_2(t_2)}$ *can not decrease to the lower value than to the level of price* $\boldsymbol{P_1(t_1)}$ *on the end of Di - phase (if it happens as a consequence of a degradation of product properties in time interval* $\boldsymbol{<t_1, t_2>}$*, than a financial doping must come from a distributor).*

e) The course of graphical dependency $\boldsymbol{VA = f(t)}$ *has a continual curves between two neighbouring OLC phases.*

Further functional dependence of the dependent variable **VA** (dimensionless) to the Real Time eliminates of time-space relativity of the Price. The **VA** function is expressed

$$VA_i = \frac{P_i}{P_{\max} - P_{\min}} = P_i \cdot \frac{1}{\Delta P} \qquad (2)$$

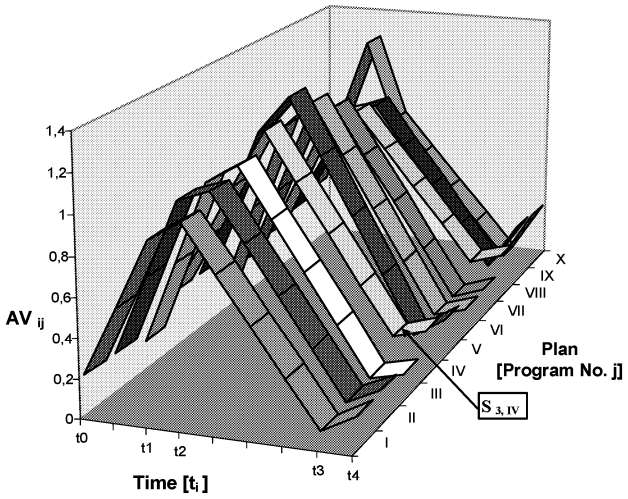
The $\mathbf{P_{max}}$ is the greatest price in OLC time interval $\mathbf{<t_0, t_4>}$, and the $\mathbf{P_{min}}$ is the lowest price here. The **ΔP** is price span for all MSW OLC phases. The **VA** is directly proportional to the price alteration at the vertical ordinate. Than can be defined discrete $\boldsymbol{\tau_i}$ as a degree of the transformation in individual phases of PLC and its relation to the time **t** will be as a first derivation of the equation (2)

$$tg\tau_{i-1,i} \equiv tg\tau_i = \frac{dVA_{i-1,i}}{dt} = VA_i' = \left(P_i \frac{1}{\Delta P}\right)' = \frac{P' \cdot \Delta P - P \cdot \Delta P'}{\Delta P^2} \qquad (3)$$

The discrete value $\mathbf{tg\,\tau_i}$ deduces a *transformation rate* $\boldsymbol{\tau_i}$ as a gradient of line $VA_{i-1,i}$ of individual PLC phase in time dependence. Prominent positive transformation $\boldsymbol{\tau_i}$ slip into both the Pr- and the Re - phases. The characteristics of a Di – phase is a possibility of positive but even negative transformation τ_i. Here the positive transformation is grown less than in the Pr–phase. The negative value of a $\boldsymbol{\tau_i}$ comes as a degradation of Object properties in the interval $\mathbf{<t_1, t_2>}$. The recycling or liquidation of an Object with negative P_3 needs substantial financial doping from a distributor and/or co-operator. The consequence of this is an obligation of a *recycling operation* in another co-operative subject environment, different from till now defined Environment. It needs considerable redundant costs.

LOGISTIC CRATE ANALYSIS

The 2D graph (parameterised by **VA – t)** is not able to be subjected optimising procedure. The third parameter must be defined as a "logistic processing Plan", which is parameterised by definite number of "logistic process Programs". It is defined at large Environment of logistic Plan and it courses at Particular Environments of logistic Programs. Then 3D mathematic graphs serve for expressions of logistic **OPERATION** (Urbánek and Rýznar 2003) by the method of **Logistic Crate Analysis** (LCA). The adjective „crate" is used for the reason that next 3D graphs have visual shape of the crate or of a cage. The substance of this method is mathematical graphical representation of time – space structured dependence of variable **VA**. The respect is necessary for above mentioned conditions a), b) c), d). But above condition e) has not validity in connection with next "jumps". The optimising criterion is "logistic processing Modality" (Urbánek 1999), which can have various criteria functions (e.g. maximum economic efficiency, minimum ecologic loading, legislative barrier etc.).



1st Graph: 3D graphic description of Logistic Crate (Plan)

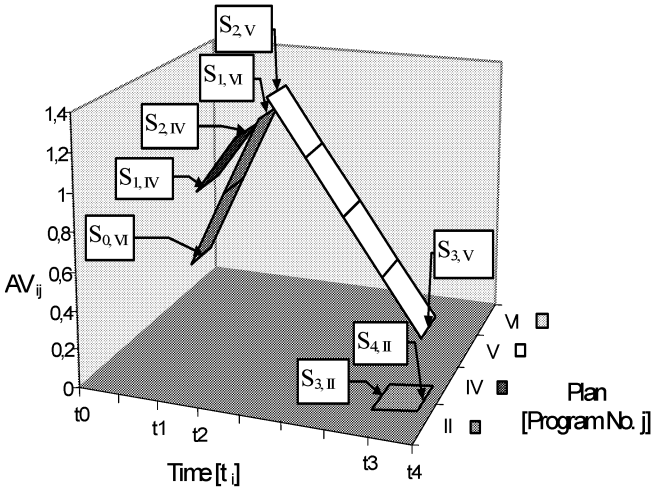
The Object space Occurrence is given with real Object position in logistic Plan at large Environment (see next 1st and 2nd Graphs of the 3D Crates). Than the separate logistic Programs (i.e. one 2D MSW OLC for simple Object) have an obligatory sequential order at separate defined Particular Environments. They can be obtained from the *control subsystem* of particular logistic Operation. These Programs follows from time-space determined logistic processes (production, storage, distribution, recycling, traffic, medicine care and so on). The Plan creation result from process of information treatment and it is given from *information subsystem* (i.e. from logistic Plan). Than, the Crate can be defined as a set of disposable program charts.

The co-ordinate **i** represents of product occurrence place at time dependence in the frame of one Program. A condition is given, that model described of the LCA must be in the right coincidence with reality. Than must be given further (third) co-ordinate **j**. The **j** characterizes next possible occurrence place of the Object by means of the realization of **j** - disposable Programs in the frame of logistic Plan. The dimension of the **j** take discontinuous values from the range j = {I., II., III, ... Z.}. Than can be defined a place of Object occurrence S_{ij}, which is parameterised by two variable t_{ij} and **j** in 3D crate

$S_{ij} = f(t_{ij}, j)$ (4) and $VA_{ij} = f(S_{ij}, t_{ij})$ (5)

The t_{ij} are Crate Nodal points where **i** = {0, 1, 2, 3, 4}.

In the frame of logistic Operation, the real Object occurrence place may be given in any Programs. Nevertheless, the real Object occurrence place must be given by the time sequential course across of concrete Programs, which are defined in the concrete Plan. The optimised logistic Programme as a result of optimised passage through the Plan is named the logistical Pipeline. The Pipeline can be defined as a realized Decision of logistic Management, which is made at the Plan by means of logistic passage through the Programs including all OLC phases. The Pipeline is shown in the 2nd Graph. The logistic management decides its Plan **α** realization and the result is Pipeline **A**. The criteria function influencing its *Decision* can be a minimization of logistical costs.



2nd Graph: Pipeline A(VI., IV., V., II). at logistic Plan α.

The Pipeline **A** courses across of **j** - Programs by means of some „jumps" between of individual Programs in time fixed Crate Nodal points t_{ij}. Than $< t_{i-1,j}, t_{i,j} >$ are constant for all **j** and here are fixed $t_{ij} = t_i$. Next condition is for simplicity here:

f) Necessary time for the jump $t_{J\Delta j}$ will be very short and is included to $< t_{i-1,\,j}, t_{i,\,j} >$ time interval and the jumps are only in Fixed Nodal time Points.

It is not possible to lost from a mind the costs – i.e. a „value-added from jumps" - ($VAJ_{\Delta j}$). Further condition is:

*g) The $VAJ_{j-1,j}$ is constant (between two neighbouring, but even between any two Programs) for all **j**.*

The **VAJ** are projected in the Object price in the reality. Here is missed out a fact that the $VAJ_{\Delta j}$ can be directly proportional to real space distances between Programs. The decision about the jumps is fully in logistical management arrangement.

CONCLUSION

The formal apparatus of DYVELOP - LCA method can be used for logistical flow optimising in medical care processes. It gives to the medicine care management quite new instrument for the optimisation of logistic costs.

REFERENCES

Dvořáková,J.; Mikša,Z. and Urbánek,J.F. 2002 "Theory of Processes – application possibilities", *The 13th INTERNATIONAL DAAAM SYMPOSIUM , "Intelligent Manufacturing & Automation: Learning from Nature",* Wien, 193-194, ISBN 3-901509-29-1

Langford,J.W. 1995. "Logistics Principles & Applications". *McGraw-Hill, Inc.USA,* ISBN 0 07 036415 X.

Mikša,Z., Hošáková,Z., Dvořáková,J., Urbánek,J. and Urbánková,V. 2003. "An Implementation of DYVELOP Method to Process System Design", *European Simulation and Modelling Conference, ESM©' 2003,* University of Naples II, Italy.

Urbánek,J.F., Jurová,M. and Dvořáková,I., 1998. "Waste Treatment in the Logistical Chains", *Proceedings of the 14thIntl Conference on Solid Waste Technology and Management,* Philadelphia, U.S.A., ISSN 1091-8043

Urbánek, J.F. 1999. "The Recycling of Waste on Multi-grade PLC Structure", *1st International Conference on Solid Waste, Technology, Safety, Environment,* Rome, Italy, ICSW –99-1803.

Urbánek, J.F. 1999. "New Instrument of Integrated Waste Management–DYVELOP", *15th International Conference on Solid Waste Technology and Management,* Philadelphia, U.S.A., ISSN 1091-8043

Urbánek,J.F. and Skála,Z. 2001. "Theory of Processes© - new Language of Human Sustainable Development", *International Containment & Remediation Technology Conference and Exhibition,* USA, Orlando.

Urbánek.J.F. 2002. „Theory of Processes – Management of Environments". *Czech language book, edition of the CERM Brno,* ISBN 80-7204-232-7. 111.

Urbánek, J.F. and Rýznar,B. 2002. "Environmental Theory of Processes", *MESM´2002 International Conference,* Sharjah, U.A.E.,. pp. 209-213, ISBN 90-77039-09-0.

Urbánek,J.F. and Urbánek,K. 2003. "The Theory of Process Relativity upgrades the Systems Concepts". *Symposium of Critical Design Issues for the Human-machine Interface,* NATO Conference, Prague.

Urbánek,J.F. and Rýznar,B. 2003. "Military Simulation Technology", *Symposium of Critical Design Issues for the Human-machine Interface,* NATO Conference, Prague.

AUTHOR BIOGRAPHY

Zuzana Hošáková was born in Zábřeh na Moravě, Czech Republic and she went to the Brno University of Technology, Faculty of Mechanical Engineering, where she obtained academic degree in 2002. Now, she is doctoral student of Brno University of Technology and she solves the tasks of process management.

THE IMPACT OF EDUCATION ON HEALTHCARE: A MALARIA AGENT-BASED SIMULATION

Fatima Rateb
J.J. Merelo
M.G. Arenas
E.T.S.I. Informática, University of Granada,
Periodista Daniel Saucedo, s/n
18071 Granada, Spain
E-mail: {fatima,jmerelo,maribel}@geneura.ugr.es

Bernard Pavard
GRIC - IRIT,
Institut de Recherche en
Informatique de Toulouse,
31062 Toulouse, France
E-mail: pavard@cict.fr

Narjes Bellamine-BenSaoud
Laboratoire RIADI,
Génies Documentiel & Logiciel,
University of Tunis,
1000 Tunis, Tunisia
E-mail: narjes.bellamine@ensi.rnu.tn

KEYWORDS
Multi-agent system, Education, Health care, Complex system, Decentralised system.

ABSTRACT

Malaria is a vector-borne disease that greatly affects social and economic development. We adopt the complex system paradigm in our analysis of the problem. Our aim is to assess the impact of education on malaria healthcare. Multi-agent systems are employed to model the spread of malaria in Haiti, where we introduce malaria education as a possible way of regulating deaths due to the parasite. We launch three experiments, each with environment modifications: 3 hospitals; 3 hospitals and 20 schools; and 5 hospitals and 20 schools. The results of running 10 simulations for each experiment show that there is a reduction in malaria deaths not only when including schools, but that in combination with increasing the number of hospitals.

INTRODUCTION

Malaria

Malaria is a vector-borne disease that greatly affects social and economic development in the world. In 1990 it was estimated that approximately 2.2 billion people were at risk of contracting the parasite, and a further 270 million were already infected. Endemic areas are characterised by 'ideal' mosquito (anopheles being the parasite vector) habitats, which are grossly where: water is present; the temperature is at least 18ºC; and there is little pollution (Baudon 2000). Many third world rural areas meet these conditions. Efforts to eradicate this deadly disease have included using DDT to minimise the vector population, and administering antimalarial drugs to susceptible people, as a prevention. However, both methods have proved only temporarily effective. The former was first adopted in the mid 1950s with a subsequent significant global decrease in mosquito population. This was soon to become a failure when a resurgence of malaria was detected as a result of anopheles developing a resistance to the insecticide (Krogstad 1996, WHO 1996). The latter prophylaxis was the use of chloroquine as an antimalarial drug. Resistance of *Plasmodium falciparum* (the more prevalent and deadly of the four existing parasite species) to chloroquine emerged due to the massive usage of the drug (Payne 1987). As a consequence, a novel way of combating this plague would have to be devised.

Complex Systems

Pavard (2002) describes a complex socio technical system to be one for which it is difficult, if not impossible to restrict its description to a limited number of parameters or characterising variables without losing its essential global functional properties. Indeed from this definition four characteristics of such a system appear: non-determinism; limited functional decomposability; distributed nature of information and representation; and emergence and self-organisation.

Simulation as a Tool for Understanding Complex Systems

The properties above show that dealing with a complex system entails dealing with the impossibility to anticipate precisely its behaviour despite knowing completely the function of its constituents. This, combined with non-linear behaviour means that it is quite problematic if not impossible to use a mathematical or statistical approach for its analysis (Bagni et al. 2002, Pavard and Dugdale 2002). It is for these reasons that computer simulations, in this study multi-agent systems (MAS), are a more viable method for exploring complex systems.

Studying complex systems through multi-agent systems has yielded useful results such as in: the evolutionary population dynamics of settlement systems in the search of emerging spatial regularities (Aschan-Leygonie 2000); demographic phenomena through its roots in individual choice behaviour and social interactions (Janssen and Martens 2001); simulations of crowd behaviour aiming to understand its dynamic and consequent control (Gomez and Rowe 2003, Hamagami et al. 2003).

Haiti

The level of poverty in Haiti reaches 65% (PAHO 2001), a socio-economic factor affecting access to public healthcare. Not only is an adequate health infrastructure not fully developed, but 'individual' poverty also hinders access to healthcare. This is aggravated further by not having the

financial resources to travel to the place of care, or not judging it necessary to seek medical care.

Malaria is considered a public health problem in Haiti (PAHO 2001), especially in rural areas. Malaria education, or its lack thereof, plays an extremely important role in the 'healing' process. It is primordial for effective and efficient treatment that malaria be diagnosed at an early stage (Baume and Kachur 1999). In order for this to apply, the population must be completely aware of its symptoms and act consequently. Symptoms which can be easily mistaken for another disease include: high fever; vomiting; convulsions; and anaemia. Not only must the population attribute specific symptoms to malaria, but they must also seek the correct medical attention. The first problem to tackle is educating the population, which could be done through national schooling. However, school attendance by children from lower income families is limited by the cost of school fees and curtailed by child labour.

STATE OF THE ART

The motivations that drive us to develop our model are various. Recent research has demonstrated approaches to the global problem, using MAS, from two angles. Janssen and Martens (1997) focus on the adaptiveness of mosquitoes to insecticides and malaria parasites to antimalarial drugs. This work aims to find a solution to controlling the spread of the disease by understanding the mechanism that renders this prophylaxis useless. Similarly, the same result is sought by Carnahan et al. (1997) but by studying the problem at a different level: the dispersal of anopheles. Here there is a focus on understanding the behaviour of malaria-transmitting mosquitoes, their geographical displacement, with the aim to consequently monitor their movements and thus reduce the number of malaria cases. Presently there is little information available showing the impact of education on healthcare in general, and even less in the tackling of malaria. We therefore attempt approaching the problem from this standpoint using MAS (StarLogo, http://www.media.mit.edu/starlogo/), more specifically applied to Haiti.

THE MODEL

Our model aims to encompass the malaria problem in Haiti. We have programmed the environment to represent the geographical terrain and the agents to represent the human population.

The Environment

The environment we create, for our agents to inhabit, is made up of a model map of Haiti, with geographical terrain granularity sufficient to represent that which affects the dynamics of what we intend to model. This granularity is such that the simulation space is divided into micro-environments: sea; hospitals; land; mountains; cities; roads; and schools. All of these micro-environments have a direct impact on our agents and hence the simulations we run.

The Mosquito Population

Our model represents only the parasite carriers of the entire anophele population, unlike in Janssen and Martens (1997). We do not model seasonal mosquito population variations. All of our modelled mosquitoes pose a malaria threat to the human (agent) population concerned.

We have decided not to model mosquitoes as an agent whose behaviour is affected by its interaction with both the environment and other agents. Their presence in the model is stochastic, embedded in the environment we create. According to conditions the seven micro-environments present, the probability of an agent contracting malaria differs. We can observe for example that 'land' (rural areas) is the ideal breeding ground for mosquitoes. This is contrary to mountains where despite the adequate water level and lack of pollution, elevation lowers ambient temperature, making it an unsuitable mosquito habitat. We consequently say that the highest probability of malaria infection is in 'land', and degressively in: road; city; and mountain. No contamination occurs in a hospital or school. This stochastic order abides to the information given on such habitats (Baudon 2000), see Table 1.

Table 1 : Mosquito Contamination Probabilities

Micro-environment	Contamination Probabilities
Land	2%
Road	1%
City	0.66%
Mountain	0.5%

Micro-environments not included are those with 0% probabilities

The Human Population

The initial human population in our model is distributed in the 5 cities in our map, with 200 agents in each. We have endowed some of our agents with the ability to be mobile, and if so, a fraction with a car. This translates into those mobile exiting their city of origin with greater ease than those not mobile. Similarly, car owners can move throughout the country at a greater speed, specially on roads, than those without a vehicle.

Natural inoculation occurs through continuous repetitive contamination, where a person cured from malaria is immune to the parasite for an average of one year (Baudon 2000). As there must be malaria-person contact, and a greater number of anopheles are found in rural areas, the initial immune and contaminated population are all mobile.

Our agents have been assigned one of the following three states: safe, when they are susceptible to contracting malaria; contaminated; and immune. Each agent can go through the malaria cycle of being safe, becoming contaminated and consequently either dying of lack of treatment or becoming cured as a consequence of a hospital visit. These states are dependent on the interaction of agents with their surrounding environment.

Baume and Kachur (1999) stress the importance of educating the population with the recognition of malaria symptoms and the gravity of not acting consequently. We have introduced this facet of the problem by creating an 'education scale' where agents have education points 1-20. Points represent the time agents take to attribute existing symptoms to malaria, where an agent with 1 education point 'waits' longer before heading towards a hospital than its counterpart with 20 points, who as soon as it is contaminated seeks medical attention. The maximum 'waiting' period is 29 days, because 30 days after contamination, an agent outside a hospital dies. Points are cumulative only. Schools are distributed throughout our Haiti map, both in rural and urban areas. The utility of a school lies in that agents moving randomly arrive at a school and leave with more malaria awareness. They enter the school, if they do not have 20 points and are not contaminated. They remain for a period of three days, after which they gain an education point.

The model emulates the contamination process stochastically through its environment. Only those agents whose state is safe can be contaminated when in a micro-environment and according to the probabilities.

Contaminated agents are to 'wait' an amount of time, depending on the education they have, as discussed above. If such agent has a maximum education of 20, the shortest distance between itself and the existing hospitals will be calculated. Subsequently the agent will start heading towards medical attention. As a contaminated agent, the speed at which it proceeds is diminished by 50% (due to weakness caused by the parasite. Those contaminated who have reached a hospital in time, will remain there for a period of 20 days, the average malaria recovery time (Malaria Foundation International 2000), and subsequently the agent's state changes to immune (during 1 year). Education's role is seen in the model when the contaminated agent, because of lack of malaria awareness, does not recognise symptoms in time and hence cannot reach a hospital. In our model death strikes when a period of 30 days has elapsed after symptoms appear, the average interval (Malaria Foundation International 2000).

The sex of an agent is not explicit. This factor only affects our model when breeding occurs. We have embodied it by using a random number generator allowing an agent to reproduce 50% of the time, as the male female ratio is approximately 1:1 in Haiti. The above condition in combination with the following must be satisfied before an agent can reproduce: minimum age of 14 years; maximum age of 49 years; not have reproduced more than 6 times; and have at least an interval of 1 year after reproduction. (WHO 2001)

As well as death caused by malaria, we have included natural deaths. The average life span for men and women in Haiti is 50.6 and 55.1 years respectively (WHO 2001). In order to accommodate these data, bearing in mind that our agent population is sex-less, we have set a maximum age of 55 years. If an agent survives malaria it dies when attaining that age.

EXPERIMENTS AND RESULTS

Our model, described attempts to encompass the present malaria situation in Haiti, in addition to information we have deemed relevant to the parasite problem. We ran simulations with 3 different scenarios: environment changes in our model. Henceforth, our 3 scenarios will be denominated in the following manner: Experiment A (3 hospitals); Experiment B (3 hospitals and 20 schools); and Experiment C (5 hospitals and 20 schools). Each experiment constitutes 10 simulations, whose duration is of 10 years.

Three Hospitals, no Schools (Experiment A)

This experiment is the basis of our comparison. We have obtained results from running simulations of our original model. Three hospitals and no schools are included in the MAS environment.

Three Hospitals, 20 Schools (Experiment B)

Our aim is to observe the effect of adding schools to the model environment. We therefore ran a further 10 simulations with 20 schools, distributed randomly in the environment. These represent malaria education initiatives that could be adopted, in order to reduce deaths.

Our hypothesis, of education having a significant positive effect on controlling malaria deaths, yielded mediocre results: not improving the present Haiti situation. Each curve in Fig. 1 is an average taken from the 10 simulations in each experiment, with corresponding standard deviations. The goal is to minimise this curve. Graphically we can note minimal difference from Experiment A to Experiment B. However, taking the area under each curve (AUC) pointed to a slight improvement with Experiment B: AUC(A)=1.30; AUC(B)=1.19.

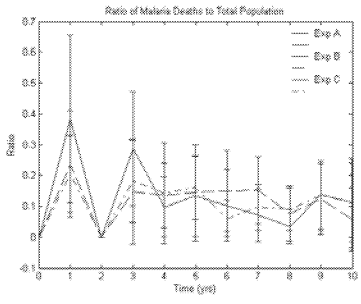


Figure 1 : Ratio of Malaria Deaths to Total Population

We found that despite a net improvement in average education as the simulation progressed (see Table 2), the malaria awareness acquired was not sufficient in decreasing malaria deaths, see Fig. 1. We attribute this to the great distances between some contaminated agents (aware of their malaria state) and the closest hospital to them. Regardless of having maximum education, the symptom appearance interval elapsed before the agent could reach medical assistance. These preliminary results drove us to experiment with increasing the number of hospitals to five, one for each city.

Table 2 : Average Education

Average Education (σ)	Initial	Final
Experiment A	9.43 (0.19)	13.84 (0.26)
Experiment B	9.55 (0.23)	19.23 (0.21)
Experiment C	9.53 (0.20)	19.27 (0.10)

We present the average education of initial and final simulation population, taken for each experiment. Standard deviations refer to the spread of the 10 experiments within each experiment group.

Five Hospitals, 20 Schools (Experiment C)

This experiment was composed of modifying further our modelled environment, by adding 2 hospitals. By doing so, distances between certain contaminated agents and a hospital are reduced, thereby increasing the possibility of them obtaining medical attention.

The ratio of malaria deaths to total population in Fig. 1, decreases in Experiment C, where AUC(C)=1.13, a lower value, as expected, than AUC(A) and AUC(B).

In order to help us have a deeper insight into the impact of education, we plot the ratio of contaminated agents in hospital with respect to the entire contaminated population, see Fig. 2 on the previous page. It exposes the proportion, for the three experiments, of contaminated agents receiving medical attention. Calculating individual experiment AUCs result in: AUC(A)=3.15; AUC(B)=3.17; AUC(C)=3.65. There is a noticeable increase from Experiment A, to B and finally C. The implication of this is explained in the Discussion section.

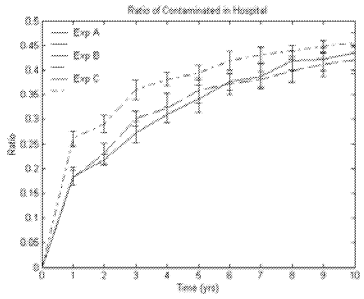


Figure 2 : Ratio of Contaminated Agents in Hospital to Entire Contaminated Population

Observing population dynamics was achieved by plotting them individually, see Figs. 3 and 4. Here the number of safe and contaminated agents is recorded so as to examine whether differences exist between such states throughout the three experiments. When analysing results for Experiments A and B in comparison to Experiment C, the immune and safe populations display a considerable increase for the latter. This, however, cannot be said for the contaminated population where we observe minimal variations between experiments, due to the fact that the factors influencing it do not vary across experiments.

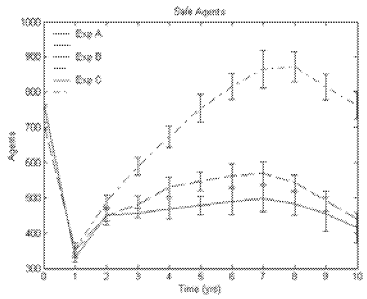


Figure 3 : Safe Agent Population

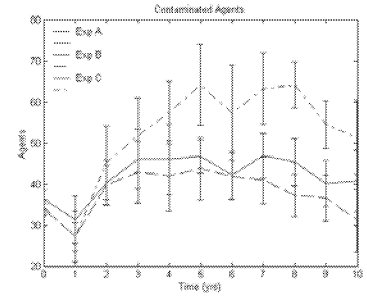


Figure 4 : Contaminated Agent Population

Individual population variations. The two figures represent averages of 10 simulation runs within each experiment, and their corresponding standard deviations.

DISCUSSION

Our results demonstrate the impact of education on malaria deaths. We have seen in Table 2 the changes in average education throughout the entire agent population for all three experiments. In Experiment A, despite the absence of schools, hence no 'learning' occurring, there was an increase in final education. This can be explained through natural selection. Those agents with a lower education level had not enough time to obtain medical assistance and consequently died. However, in the case of Experiments B and C, there is a significantly higher final average education, not only because of natural selection, but also due to the addition of schools.

The effect of education was viewed from many facets of our model. One of these is the ratio of malaria deaths to the entire population. The improvement displayed by Experiment B, in comparison to Experiment A was not as pronounced as expected. Taking the area under the curve reflected a crisper improvement, similarly to Experiment C.

Moreover, our new hypothesis (the positive impact of adding schools and hospitals), is somewhat confirmed by observing the dynamic ratio of contaminated agents in hospital with respect to the entire contaminated population, see Fig. 2. We expected to witness an increase in this ratio as the simulation progressed, whereby a greater number of contaminated agents are in a hospital obtaining medical attention. There is a clear improvement in Experiment C where its temporal variations surpass those of experiments A and B. This visual observation is confirmed when calculating AUCs for each experiment.

To corroborate the above results we can observe individual

population variations in Figs. 3 and 4. The contaminated population is very similar in all three experiments, which is to be expected as the contamination algorithm was not modified. However, the same cannot be said for the number of safe and immune agents. There is a significant increase in Experiment C in comparison to both A and B. The state cycle is such that an immune agent must be previously a contaminated agent, unless it is a child of an immune agent (children of immune agents are also immune up to 1 year after birth). This implies that those immune agents must have visited a hospital, been cured and subsequently become immune. Intuitively, we can therefore say that the increase in the immune population in Experiment C is due to a greater number of contaminated agents having sufficient education and being close enough to a hospital in order to rid themselves of the parasite.

In conclusion, we notice from the several experiment results described above that there is an improvement not only when introducing schools but also increasing the number of hospitals in our model.

FUTURE WORK

Our aim was to encapsulate epidemiological, environmental and socio-economic factors in our model. However, we would like to attempt to include greater realism in our current efforts. This would be including rural population and climatic seasons. The latter emulates the fluctuations in mosquito population and hence in probabilities of malaria contamination. Our intention is also to run simulations for a longer period.

With respect to education, a future step could be modelling the loss of education points. This could be used to capture the idea that for example after 20 years, a person can forget what it has been taught or that the treatment will have progressed, therefore information acquired previously has become obsolete.

As we have demonstrated, education was not sufficient in our model. We had to include more hospitals. This reflects the vast distances that some agents had to travel. A future step could then be to simulate the effect of improving road infrastructure and transport.

Our current model lacks realism in spatial constraints. For example with hospitals, where there is a maximum patient capacity for every hospital. This could be achieved by applying a cellular automata layer as Hamagami et al. (2003) describe.

Finally, we can say that our model could be extended to produce a generic model adaptable to different countries or geographical areas, with changes in certain parameters. Only parameter changes are needed, as the mechanisms of the malaria problem, described in the Model section, are universal. This could be the basis of an adaptable, flexible model.

ACKNOWLEDGEMENTS

Thanks go to Dr. J. E. Rowe for his input in modelling malaria contamination. This work was funded by the EU Framework 5 (DG XII TMR) project *Complexity in Social Sciences*, RTN Contract Number HPRN-CT-2000-00068. See http://www.irit.fr/COSI/ for more details.

REFERENCES

Aschan-Leygonie, C.; H. Mathian; and L. Saunders. 2000. "A Spatial Microsimulation of Population Dynamics in Southern France: a Model Integrating Individual Decisions and Spatial Constraints". In *Applications of Simulations to Social Sciences*. Ballot, G. and G. Weisbuch (Eds.). HERMES Science Publishing ltd, 109-125.

Bagni, R.; R. Berchi; and P. Cariello. 2002. "A Comparison of Simulation Models Applied to Epidemics." *Journal of Artificial Societies and Social Simulation,* 5(3), June.

Baudon, D. 2000. "Les Paludismes en Afrique Subsaharienne". *Afrique Contemporaine*, Numéro spécial, La Santé en Afrique: Anciens et nouveaux défis, 195, July-Sept. La Documentation française, 36-45.

Baume, C. and S.P. Kachur. 1999. "Améliorer la Prise en Charge Communautaire du Paludisme Infantile: Comment la Recherche Comportementale Peut-Elle Aider?". Technical report Support for Analysis and Research in Africa (SARA) Project. Academy for Educational Development (AED), United States Agency for International Development (USAID).

Carnahan, J.; S. Li; C. Costantini; Y.T. Toure; and C.E. Taylor. 1997. "Computer Simulation of Dispersal by Anopheles Gambiae in West Africa". In *Artificial Life V*. MIT Press, 387-394.

Gomez, R. and J.E. Rowe. 2003. "An Agent-Based Simulation of Crowd Movement." In *UK Simulation.* D. Al-Dabass (Eds.)

Hamagami, T.; S. Koakutsu; and H. Hirate. 2003. "A New Crowd Simulation Method by Using Multiagent on Cellular Automata" In *Artificial Life VIII.* MIT Press.

Jaeger, W. and M.A. Janssen. 2001. "Diffusion Processes in Demographic Transitions: a Prospect on Using Multi Agent Simulation to Explore the Role of Cognitive Strategies and Social Interactions". Technical report 01B40. SOM (Systems, Organisation and Management) University of Groningen.

Janssen, M.A. and W.J.M. Martens. 1997. "Modeling Malaria as a Complex Adaptive System". In *Artificial Life III.* MIT Press, 213-236.

Krogstad, D.J. 1996. "Malaria as a Reemerging Disease." *Epidemiological Reviews*, 18(1), 77-89.

Malaria Foundation International. 2000. http://www.malaria.org, June.

PAHO (Pan American Health Organization). 2001. "Basic Country Health Profiles". http://www.paho.org, May.

Pavard, B. 2002. "Complexity Paradigm as a Framework for the Study of Cooperative Systems". Lecture handouts, COSI Summer School on Modelling and Designing Complex Organisational Systems, Chania, Greece, June-July.

Pavard, B. and J. Dugdale. 2002. "An introduction to complexity in social science". Tutorial on complexity in social science, COSI project

Payne, D. 1987. "Spread of Chloroquine Resistance in Plasmodium Falciparum." *Parasitology Today*, 3, 241-246.

WHO (World Health Organization). 2001. Annex Table 2: Basic Indicators for all Member States, World Health Report 2000. http://www.who.org.

WHO (World Health Organization). 1996. "Fighting Disease Fostering Development". The World Health Report, Geneva.

ANALYTICAL AND NUMERICAL MODELLING TECHNIQUES

NUMERICAL MODELLING TECHNIQUES

DIRECT NUMERICAL METHODS OF MATHEMATICAL MODELING IN MECHANICAL STRUCTURAL DESIGN

Jihad Sahili
LRMA, Laboratoire de Recherche en Mécanique et Acoustique
University of Burgundy
49 Rue Mlle Bourgeois
B.P 31, 58027-Nevers Cedex,
France
E-mail: Jihad.Sahili_isat@u-bourgogne.fr

KEYWORDS
Mechanical engineering, Numerical methods, Validation, Solvers, Matrix.

ABSTRACT

Structural design and numerical methods are generally interactive; requiring optimization procedures as the structure is analyzed. This analysis leads to define some mathematical terms, as the stiffness matrix, which are resulting from the modeling, and then used in numerical techniques during the dimensioning procedure. These techniques and many others involve the calculation of the generalized inverse of the stiffness matrix, called also the 'compliance matrix'. The aim of this paper is to introduce first, some different existing mathematical procedures, used to calculate the compliance matrix from the stiffness matrix, then to apply direct numerical methods to solve the obtained system with the lowest computational time, and to compare the obtained results. The results show a big difference of the computational time between the different procedures.

1-INTRODUCTION

In engineering structural design, in order to achieve an optimal design, we have to modify the structure and resolve the eigenproblem repeatedly. Because of the large number of analyses required, it is very important to develop efficient reanalysis techniques, especially for structures analyzed by the finite element method (FEM), since only a small number of elements are modified in each design step (Loredo 1993).

The definition of structural static reanalysis is given by Arora (Arora 1976) as follows: find the response under static load of a structure after modifications using the original response of the structure such that the computational time of reanalysis is less than the analysis time.

Reanalysis techniques may be broadly classified into two categories: Exact and approximate methods. The exact methods generate the exact solution of the modified structure equations, and are generally efficient only when small design modifications are made (Huang 1994). For approximate methods, the accuracy of the solution and the convergence speed are two important factors, which are rarely met at the same time. A compromise must be made according to the given engineering problem. Iteration techniques are classified in the approximate methods category.

This paper is concerned with the exact method, which involves the calculation of the compliance matrix from the stiffness matrix. An alternative approach, here called "regularization", is to first compute the singular stiffness matrix then modifies it using ideas from linear algebra (Verchery 1990; Verchery 1991). Verchery has done this for the case of symmetric and positive definite matrix. The compliance matrix is then calculated from the stiffness matrix and used in the reanalysis process.

2-NUMERICAL EXAMPLES

In order to test the efficiency of the methods used for the compliance matrix calculations, two examples are chosen (Gallagher 1976):

- Beam in pure bending.
- Plate in plane stresses.

The procedure applies the finite element method on each structure to find the stiffness matrix K (Dhatt & Touzot 1984), which is calculated from the elementary stiffness matrices, corresponding to the elements resulting from the meshing of the main structure, in terms of the problem parameters.

The other calculated matrix is called rigid mode matrix R. As shown before, this matrix is constructed of the structural rigid body motion displacements. Then it can be obtained from the physical properties of the problem.

Figure 1 and figure 2 show the structures of the two treated models with their corresponding nodes and meshing, n indicates the number of elements in the beam problem (figure 1). In the plate problem, the elements are triangular and the meshing and the numbering are shown in figure 2, l and h denote the elements number in x and y directions respectively.

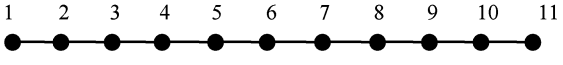


Figure 1-Beam in pure bending (n=10)

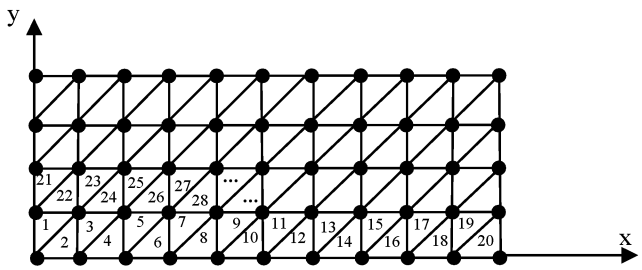


Figure 2-Plate in plane stresses (l=10, h=4)

3-RESULTS

The three methods proposed to calculate S are shown as follows:

1. $S=S_\alpha - 1/\alpha RR^t$, $S_\alpha = K_\alpha^{-1}$, $K_\alpha = K+\alpha RR^t$
2. $$KR^{-1} = \begin{bmatrix} S & R \\ R^t & 0 \end{bmatrix} = \begin{bmatrix} K & R \\ R^t & 0 \end{bmatrix}^{-1}$$
3. $$S = \begin{bmatrix} I_n \\ -A^t \end{bmatrix} \breve{S} \begin{bmatrix} I_n & -A \end{bmatrix},$$

$$\breve{S} = (I_n + AA^t)^{-1} \breve{K}^{-1} (I_n + AA^t)^{-1} \qquad R = \begin{bmatrix} R_n \\ R_r \end{bmatrix}$$

$$K = \begin{bmatrix} \breve{K} & -\breve{K}A \\ -A^t\breve{K} & A^t\breve{K}A \end{bmatrix}, \quad \breve{R} = \begin{bmatrix} R_n R_r^{-1} \\ I_r \end{bmatrix} \equiv \begin{bmatrix} A \\ I_r \end{bmatrix}$$

The third method has some advantages and disadvantages. The advantage is that the matrix to be inverted in this method is $\breve{K}$ This matrix has a banded shape and it is symmetric positive definite. The other matrix to inverse in this method is $(I_n + AA^t)^{-1}$ But this inverse has a simplified form, which involve only the inverse of an $r \times r$ matrix.

The disadvantage is coming from the multiplication of the three matrices (Golub & Van Loan 1996). This operation is very time consuming. To prevent this multiplication, it is possible to calculate the elements of $\breve{S}$, and then S only needed to treat the problem. For example, in the reanalysis relation, the matrix S is multiplied by the boundary condition matrix C, which contains only a small number of elements. This leads to define and calculate only the needed elements of $\boldsymbol{S}$. in this case; the multiplication of three matrices is not required. It is important to say that the calculation of one element of S is $O(2n^2)$ but the calculation of the global S is $O(4n^3)$.

In the following, the results are shown of the three methods. When for the third method, the total time of S calculation is shown first (figure 3-a and b), and second only the time of $\breve{K}^{-1}$ calculation is shown (Figure 3-c and d), this case is used in the reanalysis process.

The obtained results for matrix dimensions, which are increased from 10×10 up to 300×300 shows the advantage of the second method (see figure 3-a and b). The curves of the first method and third method have a parabolic shape, so for huge dimensions, these methods may be very time consuming. Instead, the second method curve has a quasi-linear shape, and a very big difference can be estimated when we compare for large dimensions. In the other hand, the geometrical and physical properties of the structure affect the numerical results only in the second method. For more complicated structure, the comparison of the figure 3-a to the figure 3-b for the second method shows the increase of the computing time of the compliance matrix. The first and the third method are remained almost the same.

For the third method, the comparison shows that if we avoid the multiplication of the three matrices, according to the problem conditions, the third method become very interesting and sometime slightly better than the second method (figure 3 c and d).

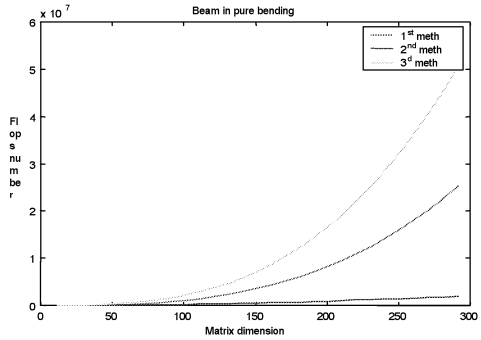


figure 3-a

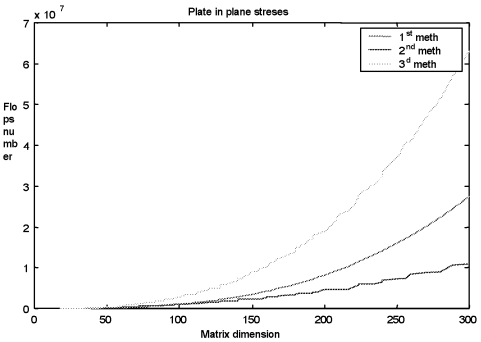


figure 3-b

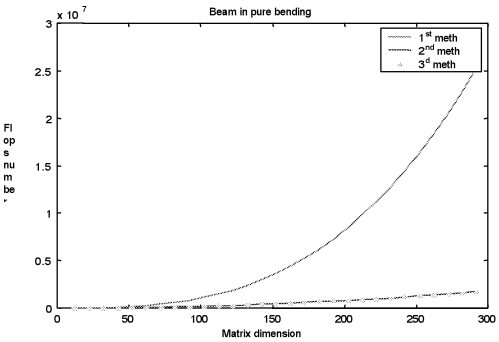


figure 3-c

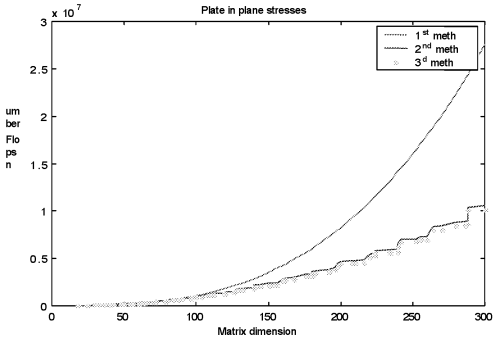


figure 3-d

Figure 3-Numerical results : a- beam in pure bending, b- plate in plane stresses. c-Beam in pure bending Calculation is limited to $\breve{K}^{-1}$ *in the third method. d-Plate in plane stresses Calculation is limited to* $\breve{K}^{-1}$ *in the third method*

4-CONCLUSION

In this paper, the three-presented mathematical procedures for the compliance matrix calculations are applied on two structures: beam in pure bending and plate in plane stresses. Results are compared between the different procedures, the type or the size of the problem. Only direct numerical methods are used (Lascaux & Théodor 1986; Meurant 1999; Muller 1989). These kinds of results could be invested in the structural design process, because problems with large and complicated stiffness matrix are often treated, and an optimized calculation method can reduce the designing time. The extensions of this work will consider problems more complicated and matrices with very big sizes; also it will be useful to try another numerical methods such as iterative methods and acceleration techniques. The parallel calculation can be used also in the development work of optimization. The parallel methods can be applied on the direct methods, iterative methods and also on some FEM code designed to construct the stiffness matrix during the designing process.

5-ACKNOWLEDGEMENT

This work is supported by the Lebanese National Council for Scientific Research

6-REFERENCES

Arora J. S. 1976. "Survey of structural reanalysis." *J. Struct. Div., ASCE*, 102, 783-802.

Dhatt G. & G. Touzot 1984. "*Une présentation de la méthode des éléments finis.*" Maloine S. A. éditeur.

Gallagher R. H. 1976. "*Introduction Aux Eléments finis.*" Pluralis, New Jersey.

Golub G. H. & C. F. Van Loan 1996. "*Matrix Computations.*" Johns Hopkins, Baltimore and London, Third Edition.

Huang C. 1994. "Méthode de réanalyse appliquée à la conception des structures composites." *Thèse de doctorat,* Université de Bourgogne.

Lascaux P. & R. Théodor 1986. "*Analyse numérique matricielle appliquée à l'art de l'ingénieur.*" Tome 1, Masson.

Loredo A. 1993. "Réanalyse des structures-Applications au contact élastique et aux matériaux incompressibles." *Thèse de doctorat*, Université Claude Bernard, Lyon I.

Meurant G. 1999. "*Computer solution of large linear systems.*" North Holland, Amsterdam.

Muller J. M. 1989. "*Arithmétique des ordinateurs.*" Masson, Paris.

Verchery G. 1990. "Régularisation du système de l'équilibre des structures élastiques discrètes." *C.R. Acad. Sci.*, 311(série II): 585-589.

Verchery G. 2-5 April 1991. "Solving structural equilibrium without boundary Conditions." (*Europian Conference on New advances in Computational structural Mechanics, Giens-France*).

JIHAD SAHILI was born in Tyre, Lebanon and went to the faculty of Engineering in the Lebanese University, where he studied the mechanical engineering and obtained his degrees in 1991. After doing an engineering training in France and Italy on industrial equipments, he worked for 5 years for industrial companies in Lebanon before moving in 1997 to the technical teaching, and in 2000, he has started his PhD in mechanics between the Lebanese University in Lebanon and the University of Burgundy in France

ADVANCED EQUATION ASSEMBLING TECHNIQUES FOR NUMERICAL SIMULATORS

Stephan Wagner[1], Tibor Grasser[1], Claus Fischer*, and Siegfried Selberherr[2]
[1]Christian-Doppler-Laboratory for TCAD in Microelectronics at the Institute for Microelectronics
[2]Institute for Microelectronics, Technical University Vienna
A-1040 Vienna, Austria
Email: Wagner@iue.tuwien.ac.at
*Firma Dr. Claus Fischer
A-2201 Gerasdorf bei Wien, Austria

KEYWORDS
Partial differential equations, Numerical methods, Computer Aided Design (CAD), Differential equation solvers, Software engineering

ABSTRACT

We present advanced equation assembling techniques as demanded by various kinds of numerical simulators solving a discretized system of nonlinear partial differential equations. Since the nonlinear problem is usually solved by a damped Newton algorithm, for each iteration one linear equation system has to be solved. The assembly approach itself is supplemented by several concepts required by the simulation process, for example the treatment of boundary conditions, physically motivated variable transformation, and numerical conditioning. The complete set of features, which has been implemented and coupled to the general purpose device and circuit simulator MINIMOS-NT, is presented in this work.

INTRODUCTION

The Finite Boxes discretization method is employed in various kinds of numerical tools and simulators for the fast and accurate solution of nonlinear partial differential equation (PDE) systems. The resulting discretized problem is then usually solved by damped Newton iterations (Deuflhard 1974, Bank and Rose 1981) which require the solution of a linear equation system at each step. The extensibility and efficiency of any simulator highly depends on the capabilities of the core modules responsible for handling the linear equation systems.

We present an advanced approach for designing the equation assembly process which has been implemented in the general purpose device and circuit simulator MINIMOS-NT (IuE 2002). Besides the basic semiconductor equations (Selberherr 1984), several different types of transport equations can be solved. Among these are the hydrodynamic equations which capture hot-carrier transport, the lattice heat flow equation to cover thermal effects like self-heating, and the circuit equations to connect single devices to a circuit, both electrically and thermally. Furthermore, various interface and boundary conditions are taken care of, which include Ohmic and Schottky contacts, and thermionic field emission over and tunneling through various kinds of barriers. This demands a sophisticated system handling the equation assembly in order to keep the simulator design flexible. To implement such a system, the requirements have been identified and generalized. A crucial aspect is also the demand for assembling and solving complex-valued linear equation systems. For that reason the module has been designed to handle both real-valued and complex-valued contributions and systems.

THE ANALYTICAL PROBLEM

In order to analyze the electronic properties of an arbitrary semiconductor structure under all kinds of operating conditions, the effects related to the transport of charge carriers under the influence of external fields must be modeled. In MINIMOS-NT carrier transport can be treated by the drift-diffusion and the hydrodynamic transport models.

Both models are based on the semiclassical Boltzmann transport equation which is a time-dependent partial integro-differential equation in the six-dimensional phase space. By the so-called method of moments this equation can be transformed in an infinite series of equations. Keeping only the zero and first order moment equations (with proper closure assumptions) yields the basic semiconductor equations (drift-diffusion model).

These equations as given first by VanRoosbroeck (VanRoosbroeck 1950) are the Poisson equation (1), the continuity equations for electrons (2) and holes (3) including a drift and diffusion term:

$$\operatorname{div}(\varepsilon \cdot \operatorname{grad} \psi) = -\rho \tag{1}$$

$$\operatorname{div}\left(D_n \cdot \operatorname{grad} n - \mu_n \cdot n \cdot \operatorname{grad} \psi\right) = R + \frac{\partial n}{\partial t} \tag{2}$$

$$\operatorname{div}\left(D_p \cdot \operatorname{grad} p + \mu_p \cdot p \cdot \operatorname{grad} \psi\right) = R + \frac{\partial p}{\partial t} \tag{3}$$

The unknown quantities of this equation system are the electrostatic potential ψ, and the electron and hole concentrations n and p, respectively. ε is the dielectric permittivity of the semiconductor, ρ denotes the space charge density, D_n and D_p are the diffusion coefficients, μ_n and μ_p stand for the carrier mobilities, and R describes the net recombination rate.

These variables depend on the unknown quantities ψ, n, and p (Selberherr 1984) and have to be modeled properly (Snowden 1989). The equation system is rendered by these models in a nonlinear form.

The heat flow equation (4) is added to account for thermal effects in the device:

$$\mathrm{div}(\kappa_\mathrm{L} \cdot \mathrm{grad} T_\mathrm{L}) = \rho_\mathrm{L} \cdot c_\mathrm{L} \tag{4}$$

This equation requires proper modeling of the thermal conductivity κ_L, the mass density ρ_L, and the heat capacity c_L. The parameters of equations (1) to (3) depend also on the lattice temperature T_L and have to be modeled properly.

Considering two additional moments gives the hydrodynamic model (Grasser et al. 2003), where the carrier temperatures are allowed to be different from the lattice temperature. Since the current densities depend then on the respective carrier temperature, two more quantities, the electron temperature T_n and the hole temperature T_p, are added.

Basically, a device structure can be divided into several segments to enable simulation of advanced heterostructures and to properly account for all conditions (which may cause very abrupt changes) at the contacts and interfaces between these segments, respectively. Every segment represents an independent domain D in one, two, or three dimensions where the PDEs are posed. The equations are implicitly formulated for a quantity x as $f_{(x)} = 0$ and termed control functions. In order to fully define the mathematical problem, suitable boundary conditions for contacts, interfaces, and external surfaces have to be applied.

Generally, such a system cannot be solved analytically, and the solution must be calculated by means of numerical methods. This approach normally consists of three tasks: at the beginning the domain D is partitioned into a finite number of subdomains D_i, in which the solution can be approximated with a desired accuracy. Then, the PDE system is approximated in each of the subdomains by algebraic equations. The unknown functions are approximated by functions with a given structure. Hence, the unknowns of the algebraic equations are approximations of the continuous solutions at discrete grid points in the domain. Thus, generally a large system of nonlinear, algebraic equations is obtained with unknowns comprised of approximations of the unknown functions at discrete points. The third task is to derive a solution of the unknowns of the nonlinear algebraic system. The quality of the approximation depends on the resolution of the partitioning into subdomains as well as on the suitability of the approximating functions.

THE DISCRETIZED PROBLEM

For the derivation of the discrete problem several methods can be applied. We deal here with point residual methods: the finite difference method based on rectangular grids or the finite boxes (box integration) method allowing general unstructured grids.

Nonlinear partial differential equations of second order can appear in three variants: elliptic, parabolic, and hyperbolic PDEs. The Poisson equation as well as the steady-state continuity equations form a system of elliptic PDEs, whereas the heat-flow equation is parabolic. To completely determine the solution of an elliptic PDE boundary conditions have to be specified. Since parabolic and hyperbolic PDEs describe evolutionary processes, time normally is an independent variable and an initial condition is additionally required. Hence, also the transient continuity equations are parabolic.

Applying the finite boxes discretization scheme (Selberherr 1984) the equations are integrated over a control volume (subdomain, usually obtained by a Voronoi tesselation) D_i which is associated with the grid point P_i. For this grid point a general equation for the quantity x is implicitly given as

$$f^\mathrm{S}_{x_i} = \sum_j F_{x_{i,j}} + G_i = 0 \tag{5}$$

where j runs over all neighboring grid points in the same segment, $F_{x_{i,j}}$ is the flux between points i and j, and G_i is the source term (see Fig. 1). Grid points on the boundary ∂D are defined as having neighbor grid points in other segments. Thus, (5) does not represent the complete control function $f_{(x)}$, since all contributions of fluxes into the contact or the other segment are missing. For that reason, the information for these boxes has to be completed by taking the boundary conditions into account. Common boundary conditions are the Dirichlet condition which specifies the solution on the boundary ∂D, the Neumann condition which specifies the normal derivative, and the linear combination of these conditions giving an intermediate type:

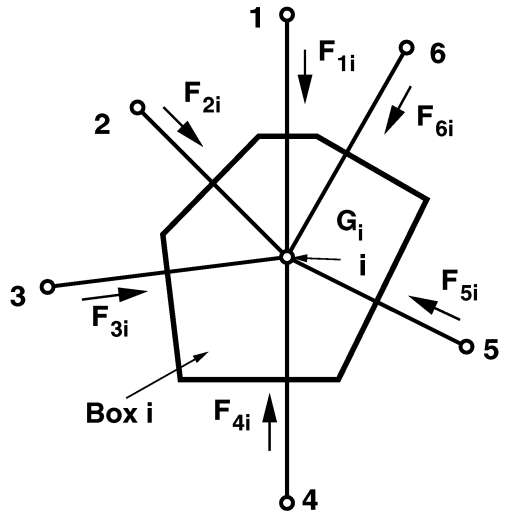


Figure 1: Box i with 6 neighbors

$$\mathbf{n} \cdot \mathrm{grad} x + \sigma x = \delta \tag{6}$$

Generally, the form of these conditions depends on the respective boundary models. For that reason the equation assembly is often performed in a coupled way, causing complicated modules. For instance, it is absolutely necessary to differ between interior and boundary points. Considering a general tetrahedron, there exist many kinds of boundary points

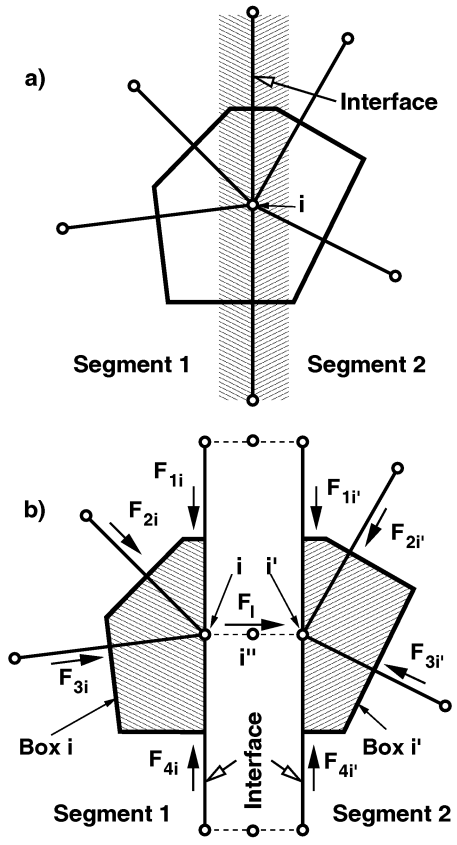


Figure 2: Splitting of interface points: Interface points as given in a) are split into three different points having the same geometrical coordinates b)

(depending on the number of edges involved), which have to be treated separately. This leads to a complicated implementation of the models and can make simplifications necessary. Thus, due to organizational and implementational issues this form of coupling should be avoided.

More complex models with exponential interdependence between the solution variables such as thermionic field emission interface conditions (Schroeder 1994) have also been implemented.

The method which has been developed allows to implement the segment models which describe the interior fluxes and their derivatives independently from the boundary models. The segment models do not have to differentiate the point type, they do not even have to care about the boundary model used. The assembly system is responsible for combining all relevant contributions by using the information given by the boundary models.

Interface Conditions

To account for complex interface conditions, grid points located at the boundary of the segments have three values (see Fig. 2), one for each segment and a third point located directly at the interface which can be used to formulate more complicated interface conditions, e.g. like interface charges. However, to simplify notation these interface values will be omitted in our discussion and only the two interface points, i and i', are used. Basically, the two equations $f^{\mathrm{S}}_{x_i}$ and $f^{\mathrm{S}}_{x_{i'}}$ are completed by adding the missing boundary fluxes $F_{x_{i,i'}}$:

$$f_{x_i} = f^{\mathrm{S}}_{x_i} + F_{x_{i,i'}} = 0 \quad (7)$$

$$f_{x_{i'}} = f^{\mathrm{S}}_{x_{i'}} - F_{x_{i,i'}} = 0 \quad (8)$$

The intermediate type of interfaces (6) and thus also the two other types of interfaces are generally given in linearized form by:

$$\alpha(x_i - \beta x_{i'} + \gamma) = F_{x_{i,i'}} \quad (9)$$

α, β, and γ are linearized coefficients, $F_{x_{i,i'}}$ represents the flux over the interface. The three types of interfaces differ in the magnitude of α.

In the case of an arbitrary splitting of a homogeneous region into different segments, the boundary models have to ensure that the simulation results remain unchanged. By adding (8) to (7), the box of grid point P_i can be completed and the boundary flux is eliminated. The merged box is now valid for both grid points, for that reason the respective equation can not only be used for grid point P_i, but also for $\mathrm{P}_{i'}$.

Whereas the segment models assemble the so-called segment matrix, the interface models are responsible for assembling and configuring the interface system consisting of a boundary and special-purpose transformation matrix. New equations based on (9) can be introduced into the boundary matrix without any limitations on α, thus from 0 (Neumann) to ∞ (Dirichlet). The interface models are also responsible for configuring the transformation matrix to combine the segment and boundary matrix correctly. Depending on the interface type there are two possibilities:

- Dirichlet boundaries are characterized by $\alpha \rightarrow \infty$. Thus, the implicit equation $x_i = \beta x_{i'} - \gamma$ can be used as a substitute equation. As these equations are normally not diagonally dominant they have a negative impact on the condition number of the system matrix and are configured to be preeliminated (see below).
- For the other types (explicit boundary conditions) the boundary flux is simply added to the segment fluxes. In the case of a large α the transformation matrix can be used to scale the entries by $1/\alpha$ because of the preconditioner used in the solver module.

Note, that all interface-dependent information is administrated by the respective interface model only.

As an additional feature the transformation matrix can be used to calculate several independent boundary quantities by combining the specific boundary value with the segment entries (also in the case of Dirichlet boundaries). For example, the dielectric flux over the interface is calculated as $\sum_i f^{\mathrm{S}}_{x_i}$ and introduced as a solution variable because some interface models require the cross-interface electric field strength to

determine tunnel processes. Calculation of the normal electric field is thus trivial. Note, that this is not the case when the normal component of the electric field $\vec{E}_n$ has to be calculated using neighboring points in the case unstructured two- or three-dimensional grids are used.

Fig. 3 illustrates these concepts. The transformations are set up to combine the various segment contributions with the boundary system.

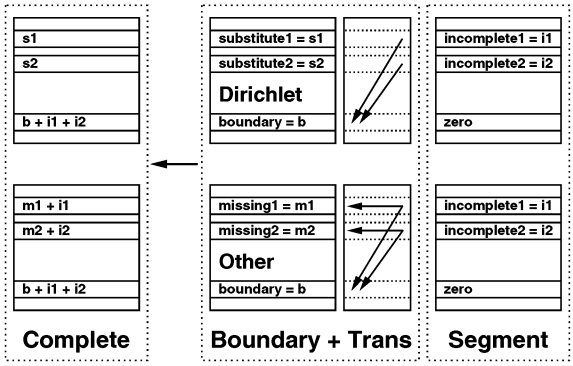


Figure 3: The complete equations are a combination of the boundary and the segment system. This combination is controlled by the transformation matrix and depends on the interface type.

Boundary Conditions

Contacts are handled in a similar way to interfaces. However, in the contact segment there is only one variable available for each solution quantity (x_C). Note, that contacts are represented by spacial multi-dimensional segments.

Furthermore, all fluxes over the boundary are handled as additional solution variables F_C (e.g., contact charge Q_C for Poisson equation, contact electron current $I_{n\mathrm{C}}$ for the electron continuity equation, or H_C as the contact heat flow).

With i running over all segment grid points, for explicit boundary conditions one gets

$$\begin{aligned} f_{x_i} &= f^\mathrm{S}_{x_i} + F_{x_i,\mathrm{C}} &= 0 \\ f_{F_\mathrm{C}} &= F_\mathrm{C} + \textstyle\sum_i f^\mathrm{S}_{x_i} &= 0 \end{aligned}$$

For Dirichlet boundary conditions one gets

$$f_{x_i} = x_\mathrm{C} - h(x_i) = 0 \qquad (10)$$

$$f_{F_\mathrm{C}} = F_\mathrm{C} + \textstyle\sum_i f^\mathrm{S}_{x_i} = 0 \qquad (11)$$

Here, x_C in (10) is the boundary value of the quantity, which is a solution variable, whereas (11) is used as constitutive relation for the actual flow over the boundary F_C. $h(x_i)$ denotes the substitute equation.

For Neumann boundaries the flux over the boundary is zero hence the equation assembled by the segment model is already complete.

Assembly of the Complete System

The semiconductor device is divided into several segments that are geometrical regions employing a distinct set of models. The implementation of each model is completely independent from other models and each model is basically allowed to enter its contributions to the linear equation system. All boundary and interface issues are completely separated from the general segment models. Hence, also completely independent assembly structures for the boundary and segment system are used.

Thus, the system matrix $\mathbf{A}$ (the Jacobian matrix in a Newton approximation) will be assembled from two parts, namely the direct part $\mathbf{A}_\mathrm{b}$ (boundary models) and the transformed part $\mathbf{A}_\mathrm{s}$ (segment models). The latter is multiplied by the row transformation matrix $\mathbf{T}_\mathrm{b}$ from the left before contributing to the system matrix $\mathbf{A}$. The right hand side vector $\mathbf{b}$ is treated the same way:

$$\begin{aligned} \mathbf{A} &= \mathbf{A}_\mathrm{b} + \mathbf{T}_\mathrm{b} \cdot \mathbf{A}_\mathrm{s} \\ \mathbf{b} &= \mathbf{b}_\mathrm{b} + \mathbf{T}_\mathrm{b} \cdot \mathbf{b}_\mathrm{s} \\ \mathbf{A} \cdot \mathbf{x} &= \mathbf{b} \end{aligned}$$

Although in principle every model is allowed to add entries to all components, the assembly module checks two prerequisites before actually entering the value: first, the quantity the value belongs to is marked to be solved (the user may request only a subset of all provided models) and secondly the priority of the model is high enough to modify the row transformation properties. As stated before, the row transformation is used to complete missing fluxes in boundary boxes. Since a grid point can be part of more than two segments, a ranking using a priority has been introduced. For example, contact models have usually the highest priority and thus their contributions are always used for completion.

All three matrices $\mathbf{A}_\mathrm{b}$, $\mathbf{A}_\mathrm{s}$, and $\mathbf{T}_\mathrm{b}$ and the two vectors $\mathbf{b}_\mathrm{b}$ and $\mathbf{b}_\mathrm{s}$ may be assembled simultaneously, so no assembly sequence must be adhered to. In addition, a forth matrix $\mathbf{T}_\mathrm{v}$ is assembled which contains information for an additional variable transformation.

THE ASSEMBLY MODULE

MINIMOS-NT consists of two separate modules responsible for assembling and solving linear equation systems. First, the assembly module which is directly accessed by the implemented physical models of the simulator, provides an effective application programming interface, various transformation algorithms and the preelimination system. In addition, sorting and scaling plug-ins can be called. Second, the solver module which is plugged into the assembly module, is responsible for solving the so-called inner linear equation system. The module currently used provides a direct (Gaussian) method and two iterative solver schemes.

The key demands on the assembly module (class) can be summarized as follows:

1. The Application Programming Interface provides methods for
   - adding values to the segment system
   - adding values to the boundary system
   - adding values to the transformation matrix
   - deleting equations
   - setting elimination flags
   - administration of priority information
2. The row transformation performs a linear combination of rows to extinguish large entries.
3. The variable transformation is used to reduce the coupling of the semiconductor equations. Especially in the case of mixed quantities in the solution vector, a variable transformation is sometimes helpful to improve the condition of the linear system. The representation chosen here allows to specify fairly arbitrary variable transformations to be applied to the system. Basically, a matrix $\mathbf{T}_{\mathrm{v}}$ is assembled and multiplied with the system matrix.
4. The preelimination is required to eliminate problematic equations by Gaussian elimination in order to improve the condition of the inner system matrix. Matrix $\mathbf{A}_{\mathrm{s}}$ consists of fluxes that will (if the control functions are correctly assigned to the variables) satisfy the criterion of diagonal-dominance that is necessary to make the linear equation system solvable with an iterative solver. The transformations and additional terms imposed by the boundary conditions may heavily disrupt this feature both in structural and numerical aspects. Some of the boundary or interface conditions can make the full system matrix so ill-conditioned that this simply prevents iterative linear solvers from converging.
5. Specific plug-ins are called for
   - Scaling: Since a threshold value (tolerance) is used to decide whether to keep or skip an entry, the preconditioner used (Incomplete-LU factorization) requires a system matrix having entries of the same order of magnitude.
   - Sorting: Reduction of the bandwidth of a matrix to reduce the fill-in.
   - Solving: Calculate the solution vector of the linear equation system.
6. After reverting all transformations and backsubstituting the preeliminated equations, the output of the assembly module is the complete solution vector. In addition, the right-hand-side vector is returned which can be used for various norm calculations.

CONCLUSION

We presented advanced equation assembly techniques which are successfully applied in the device and circuit simulator MINIMOS-NT. Among these are all features required for the effective and efficient assembling of linear equation systems. We developed a formulation which allows to independently treat segments, boundaries, and interface models. All fluxes over boundaries are available as solution variables, which simplifies the formulation of boundary conditions and circuit equations.

The presented concepts result in superior stability of MINIMOS-NT without restricting model implementation and further development. The general approach for treating boundary conditions yields in combination with several preconditioning measures diagonal-dominant linear equation systems well prepared for advanced solver algorithms. As a result, boundary conditions for specific operating points can be directly applied without successively stepping to the desired value as is very common even in commercial simulators.

AUTHOR BIOGRAPHY

STEPHAN WAGNER was born in Vienna, Austria, in 1976. He studied electrical engineering at the Technical University Vienna, where he received the master degree of "Diplomingenieur" in 2001. As a member of the MINIMOS-NT development group he joined the Institute for Microelectronics in November 2001, where he is currently working on his doctoral degree. His scientific interests include device and circuit simulation, numerical aspects and software technology in general.

REFERENCES

Bank, R.E. and D.J. Rose. 1981. "Global Approximate Newton Methods". *Numer.Math.*, 37, 279–295.

Bløtekjær, K. 1970. "Transport Equations for Electrons in Two-Valley Semiconductors". *IEEE Trans.Electron Devices*, ED-17(1), 38–47.

Deuflhard, P. 1974, "A Modified Newton Method for the Solution of Ill-Conditioned Systems of Nonlinear Equations with Application to Multiple Shooting". *Numer.Math.*, 22, 289–315.

Fischer, C. 1994. *Bauelementsimulation in einer computergestützten Entwurfsumgebung*. Dissertation, Technische Universität Wien.

Grasser, T.; T.-w. Tang; H. Kosina; and S. Selberherr. 2003. "A Review of Hydrodynamic and Energy-Transport Models for Semiconductor Device Simulation". *Proc.IEEE*, 91(2), 251–274.

http://www.iue.tuwien.ac.at/software/minimos-nt. 2002. Minimos-NT 2.0 User's Guide, IμE. Institut für Mikroelektronik, Technische Universität Wien, Austria.

Scharfetter, D.L. and H.K. Gummel. 1969. "Large-Signal Analysis of a Silicon Read Diode Oscillator". *IEEE Trans.Electron Devices*, 16(1), 64–77.

Schroeder, D. 1994. *Modelling of Interface Carrier Transport for Device Simulation*. Springer.

Selberherr, S. 1984. *Analysis and Simulation of Semiconductor Devices*. Springer, Wien–New York.

Snowden, Ch.M. 1989. *Semiconductor Device Modelling*. Springer.

VanRoosbroeck, W.V. 1950. "Theory of Flow of Electrons and Holes in Germanium and Other Semiconductors". *Bell Syst.Techn.J.*, 29, 560–607.

ON SOLVING ORDINARY DIFFERENTIAL EQUATION SYSTEMS WITH GENERALIZED STOCHASTIC PETRI NETS

Olli-Matti Penttinen
Helsinki University of Technology
Laboratory for Theoretical Computer Science
P.O. Box 9700, FIN-02015 TKK, Finland
Email: Olli-Matti.Penttinen@luukku.com

ABSTRACT

A novel method of solving differential equations with Generalized Stochastic Petri Nets is presented. With this method, called Stochastic Differential Analysis (SDA), systems of ordinary linear differential equations can be numerically solved. SDA can also be utilized for approximating solutions of non-linear differential equation systems; and, with a discretization scheme, of partial differential equations as well. Representations of an ODE-system as a GSPN and in SDA meta language are given. Properties of the underlying stochastic process, which is a multi-dimensional birth-death process, are studied. A radioactive decay process is used as an example. It is shown that the associated SDA process has less variance than a faithful physical model of the same system.

INTRODUCTION

Traditionally, continuous physical processes have been modeled with systems of ordinary and partial differential equations. This method is cumbersome or even impossible to apply, if either the processes themselves or the equations that govern them experience abrupt changes over time; especially in case the alterations are stochastic.

Formalisms such as Stochastic Petri Nets that are based on Markovian stochastic processes have been available for performance analysis of concurrent and distributed systems for more than two decades. They work very well with discrete processes, but are not directly applicable to analysis of continuous systems.

In the light of how thoroughly these disciplines are understood—both numerically and analytically—and how widespread their use, it is remarkable how difficult it seems to be to bring these worlds together to form an expressive framework for the analysis of *hybrid systems*, i.e., stochastic processes whose state spaces are partially discrete and partially continuous. Unfortunately, many real-world engineering systems fall into this category.

Recently, several continuous and hybrid classes of Petri nets have been proposed. Of these, *Hybrid Petri Nets* (HPN) (David and Alla 1992) and *Fluid Stochastic Petri Nets* (FSPN) (Horton et al. 1998) have received the most attention. Both of them are capable of expressing precise hybrid models, and both lead to very complicated analysis algorithms. At this point, only some of their potential modeling power can be exploited for numerical analysis.

In this paper, we introduce a novel technique called *stochastic differential analysis* (SDA) for solving systems of linear ordinary differential equations (ODE) with Generalized Stochastic Petri Nets (GSPNs) (Ajmone Marsan et al. 1995). Interfacing to a discrete GSPN-model is straightforward, because continuous variables are represented as place markings and solving their associated differential equations is simulated in the form of transitions of a GSPN. SDA has been used for simulating quantum systems in (Ojala et al. 2003; Ojala and Penttinen 2003).

STOCHASTIC DIFFERENTIAL ANALYSIS

Consider a system of N ordinary linear differential equations with constant coefficients

$$\mathbf{y}' = \mathbf{f}(\mathbf{y}) = A\mathbf{y} + \mathbf{b} \iff \tag{1a}$$

$$\begin{bmatrix} y_1' \\ \vdots \\ y_N' \end{bmatrix} = \begin{bmatrix} f_1(y_1, \ldots, y_N) \\ \vdots \\ f_N(y_1, \ldots, y_N) \end{bmatrix} \tag{1b}$$

$$= \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NN} \end{bmatrix} \cdot \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix} , \tag{1c}$$

with initial values

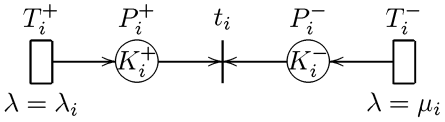
$$\mathbf{y}(0) = \mathbf{c} \iff \begin{bmatrix} y_1(0) \\ \vdots \\ y_N(0) \end{bmatrix} = \begin{bmatrix} c_1 \\ \vdots \\ c_N \end{bmatrix} , \tag{1d}$$

where $A \in \mathbb{R}^N \times \mathbb{R}^N$, $\mathbf{b} \in \mathbb{R}^N$ and $\mathbf{c} \in \mathbb{R}^N$ are constants. Further, divide each $f_i(\mathbf{y})$ into two non-negative functions $f_i^+(\mathbf{y})$ and $f_i^-(\mathbf{y})$ such that

$$f_i^+(\mathbf{y}) = \begin{cases} f_i(\mathbf{y}) & \text{if } f_i(\mathbf{y}) > 0 \\ 0 & \text{if } f_i(\mathbf{y}) \leq 0 \end{cases} , \tag{1e}$$

$$f_i^-(\mathbf{y}) = \begin{cases} 0 & \text{if } f_i(\mathbf{y}) \geq 0 \\ -f_i(\mathbf{y}) & \text{if } f_i(\mathbf{y}) < 0 \end{cases} \ . \tag{1f}$$

Each dependent variable $y_i(t)$ and the associated function $f_i(\mathbf{y})$ is modeled with a GSPN fragment



(2a)

where markings of P_i^+ and P_i^- denote positive and negative values of y_i, respectively, and t_i ensures that both places cannot simultaneously contain tokens in any tangible marking. The initial marking $M_0(P_i^+) = K_i^+$, $M_0(P_i^-) = K_i^-$ is such that $K_i^+ - K_i^- = c_i$. (We assume here that $c_i \in \mathbb{Z}$. Non-integral initial conditions are considered later.) The marking-dependent firing rates of T_i^+ and T_i^- are

$$\lambda_i = f_i^+\big((\#P_1^+ - \#P_1^-), \ldots, (\#P_N^+ - \#P_N^-)\big) \ , \tag{2b}$$
$$\mu_i = f_i^-\big((\#P_1^+ - \#P_1^-), \ldots, (\#P_N^+ - \#P_N^-)\big) \ , \tag{2c}$$

where $\#P = M(P)$, i.e., the number of tokens in place P in a marking.

Remark 1 *In each tangible marking, for each $i \in \{1, \ldots, N\}$, either T_i^+ or T_i^- or both have an effective firing rate 0, in which case they are considered not to have concession. This can easily be ensured by setting $\lambda(T_i^+) = f_i$, $\lambda(T_i^-) = -f_i$ and associating a transition* guard $\text{Guard}(T) = (\lambda(T) > 0)$ *with each timed transition.*

Definitions. For each $i \in \{1, \ldots, N\}$, let the random variable $X_i = (\#P_i^+ - \#P_i^-)$ and $\mathbf{X} = [X_1 \ldots X_N]$. Let $\{\mathbf{X}(t),\ t \geq 0\}$ be a stochastic process that is isomorphic to the tangible reachability graph of the net (2). Let $S = \mathbb{Z}^N$ denote the state space and $\mathbf{n} = [n_1 \ldots n_N] \in S$ denote a state of $\{\mathbf{X}(t)\}$ and define the probability $\pi_{\mathbf{n}}(t) = P\big[\mathbf{X}(t) = \mathbf{n}\big]$. Let the successor of $\mathbf{n}$ with respect to the ith state variable be $\mathbf{s}_i(\mathbf{n}) = [n_1 \ldots n_{i-1}\ (n_i + 1)\ n_{i+1} \ldots n_N]$ and the predecessor be $\mathbf{p}_i(\mathbf{n}) = [n_1 \ldots n_{i-1}\ (n_i - 1)\ n_{i+1} \ldots n_N]$.

Theorem 2 (Expectation of Markings) $E\big[\mathbf{X}(0)\big] = \mathbf{c} \Rightarrow \forall i \in \{1, \ldots, N\},\ \forall t \geq 0,$

$$E\big[\#P_i^+ - \#P_i^-\big]\Big|_t = E\big[X_i(t)\big] = y_i(t) \ .$$

Proof. With the above definitions, the forward Kolmogorov differential equations for state probabilities (Karlin and Taylor 1984) may be written as

$$\pi'_{\mathbf{n}}(t) = \sum_{k=1}^{N} \Big(\lambda_k\big(\mathbf{p}_k(\mathbf{n})\big)\pi_{\mathbf{p}_k(\mathbf{n})}(t) + \mu_k\big(\mathbf{s}_k(\mathbf{n})\big)\pi_{\mathbf{s}_k(\mathbf{n})}(t) - \big(\lambda_k(\mathbf{n}) + \mu_k(\mathbf{n})\big)\pi_{\mathbf{n}}(t) \Big) \ .$$

Multiplied by n_i and added together over S, they give

$$\begin{aligned}
&\sum_{\mathbf{n}\in S} n_i \pi'_{\mathbf{n}}(t) = \\
&\sum_{\mathbf{n}\in S} n_i \sum_{k=1}^{N} \Big(\lambda_k\big(\mathbf{p}_k(\mathbf{n})\big)\pi_{\mathbf{p}_k(\mathbf{n})}(t) + \mu_k\big(\mathbf{s}_k(\mathbf{n})\big)\pi_{\mathbf{s}_k(\mathbf{n})}(t) \\
&\qquad - \big(\lambda_k(\mathbf{n}) + \mu_k(\mathbf{n})\big)\pi_{\mathbf{n}}(t) \Big) = \\
&\sum_{\mathbf{n}\in S} \sum_{k\neq i} n_i \Big(\lambda_k\big(\mathbf{p}_k(\mathbf{n})\big)\pi_{\mathbf{p}_k(\mathbf{n})}(t) + \mu_k\big(\mathbf{s}_k(\mathbf{n})\big)\pi_{\mathbf{s}_k(\mathbf{n})}(t) \\
&\qquad - \big(\lambda_k(\mathbf{n}) + \mu_k(\mathbf{n})\big)\pi_{\mathbf{n}}(t) \Big) \\
&\qquad + \sum_{\mathbf{n}\in S} n_i \Big(\lambda_i\big(\mathbf{p}_i(\mathbf{n})\big)\pi_{\mathbf{p}_i(\mathbf{n})}(t) + \mu_i\big(\mathbf{s}_i(\mathbf{n})\big)\pi_{\mathbf{s}_i(\mathbf{n})}(t) \\
&\qquad - \big(\lambda_i(\mathbf{n}) + \mu_i(\mathbf{n})\big)\pi_{\mathbf{n}}(t) \Big) = \\
&\sum_{\mathbf{n}\in S} \Big(\big((n_i + 1)\lambda_i(\mathbf{n}) + (n_i - 1)\mu_i(\mathbf{n})\big)\pi_{\mathbf{n}}(t) \\
&\qquad - n_i\big(\lambda_i(\mathbf{n}) + \mu_i(\mathbf{n})\big)\pi_{\mathbf{n}}(t) \Big) = \\
&\sum_{\mathbf{n}\in S} \Big(\big(\lambda_i(\mathbf{n}) - \mu_i(\mathbf{n})\big)\pi_{\mathbf{n}}(t) \Big) = \sum_{\mathbf{n}\in S} \big(f_i(\mathbf{n})\pi_{\mathbf{n}}(t)\big) \iff \\
&\frac{\mathrm{d}}{\mathrm{d}t} E\big[X_i(t)\big] = E\big[f_i(\mathbf{X}(t))\big] \\
&= E\big[a_{i1}X_1(t) + \cdots + a_{iN}X_N(t) + b_i\big] \\
&= a_{i1}E\big[X_1(t)\big] + \cdots + a_{iN}E\big[X_N(t)\big] + b_i \ .
\end{aligned} \tag{3}$$

On the other hand,

$$y_i' = a_{i1}y_1 + \cdots + a_{iN}y_N + b \ . \tag{4}$$

The differential equations in (3) and in (4) are the same and known to have a unique solution for each initial condition. □

Remark 3 *The last step in* (3) *is possible only because each f_i is linear. In general, for non-linear functions f, the expectation $E\big[f(X)\big] \neq f\big(E[X]\big)$. Consequently,* SDA *can simulate exactly only linear* ODE *systems.*

Remark 4 *Because each state change occurs to a neighboring state with an exponentially distributed rate that only depends on the current state, $\{\mathbf{X}(t)\}$ can be considered an N-dimensional generalization of a* birth-death process.

AN EXAMPLE

To illustrate SDA, we now give a short example. We first model it with a GSPN that faithfully follows the physical reality, and then give a SDA model of the same process. It is simple enough to be solved analytically with both

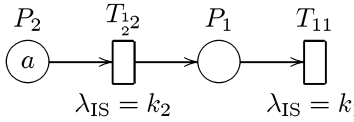
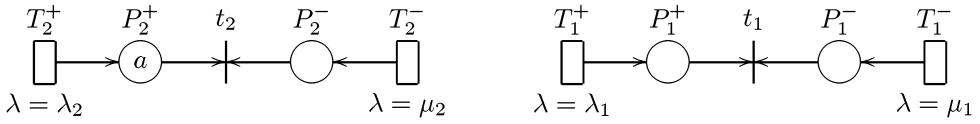


Figure 1: GSPN model of 2-phase decay



$$\begin{aligned}
\lambda_1 &= -k_1(\#P_1^+ - \#P_1^-) + k_2(\#P_2^+ - \#P_2^-) & \quad \mathrm{Guard}(T_1^+) &= \lambda_1 > 0 \\
\mu_1 &= k_1(\#P_1^+ - \#P_1^-) - k_2(\#P_2^+ - \#P_2^-) & \quad \mathrm{Guard}(T_1^-) &= \mu_1 > 0 \\
\lambda_2 &= -k_2(\#P_2^+ - \#P_2^-) & \quad \mathrm{Guard}(T_2^+) &= \lambda_2 > 0 \\
\mu_2 &= k_2(\#P_2^+ - \#P_2^-) & \quad \mathrm{Guard}(T_2^-) &= \mu_2 > 0
\end{aligned}$$

Figure 2: SDA model of 2-phase decay

methods, which enables us to compare results in closed form.

Example 5 (Two-phase Radioactive Decay) Consider a nuclear process, in which a number of some unstable nuclei, say, 'A' particles decay into intermediate, also unstable 'B' particles; and, finally, into stable 'C' particles according to the nuclear chain reaction

$$A \xrightarrow{k_2} B \xrightarrow{k_1} C \ .$$

It is our task to find out the numbers of 'A' and 'B' particles as a function of time, given that the initial amount of 'A' particles is a, there are initially no 'B' or 'C' particles; and the half-lives of 'A' and 'B' are known to be $T_{1/2}(A) = (\ln 2)/k_2$ and $T_{1/2}(B) = (\ln 2)/k_1$, respectively.

From physics, the solution of the ODE system

$$\begin{cases} y_1' = -k_1 y_1 & +k_2 y_2 \,, & y_1(0) = 0 \\ y_2' = & -k_2 y_2 \,, & y_2(0) = a \ , \end{cases} \tag{5}$$

where $y_2(t)$ is the number of 'A' particles and $y_1(t)$ is the number of 'B' particles at time t, is known to give the average numbers of particles. The solution to (5) is

$$\begin{aligned} y_1 &= \begin{cases} akt\mathrm{e}^{-kt} \,, & k_1 = k_2 = k \\ \dfrac{ak_2}{k_1 - k_2}\left(\mathrm{e}^{-k_2 t} - \mathrm{e}^{-k_1 t}\right), & k_1 \neq k_2 \end{cases} \\ y_2 &= a\mathrm{e}^{-k_2 t} \ . \end{aligned} \tag{6}$$

The process can in a very natural manner be modeled with the net in Fig. 1. Place P_2 models 'A' particles and place P_1 'B' particles. Note, that both exponential transitions $T_{1_2^2}$ and T_{11} have infinite-server semantics. Thus, their effective firing rates are $k_2 \cdot \#P_2$ and $k_1 \cdot \#P_1$, respectively. As the reader may be able to see, $\#P_2(t) \sim \mathrm{Bin}\left(a, \mathrm{e}^{-k_2 t}\right)$ and $\#P_1(t) \sim \mathrm{Bin}\left(a, k_2\mathrm{e}^{-k_2 t} * \mathrm{e}^{-k_1 t}\right)$, where the operator $*$ denotes the *convolution integral*. The convolution of functions $f(t)$ and $g(t)$ is defined as

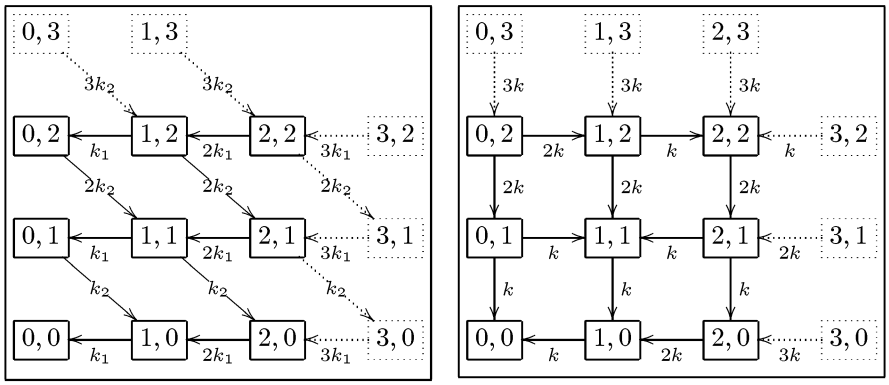
$$(f * g)(t) = \int_0^t f(\tau) g(t - \tau)\, \mathrm{d}\tau \ .$$

A SDA-model of the process is depicted in Fig. 2. (Transition T_2^+ with its annotations and marking subexpressions $\#P_i^-$ are superfluous, but included here for simplicity of exposition.)

Tangible reachability graphs of both nets are finite. The net in Fig. 1 has $(a^2 + 3a + 2)/2 = \mathcal{O}(a^2)$ and the net in Fig. 2 has $a^2 + 2a + 1 = \mathcal{O}(a^2)$ reachable tangible markings. Thus, for modest values of a, the distributions of $\#P_i^{\pm}(t)$ can be numerically computed with transient analysis of the respective underlying CTMC depicted in Fig. 3. The expected values and also the variances of the place markings can be obtained in closed form with the technique used in the proof of Theorem 2. With formulae (6) and $\#P_1^-(t) = \#P_2^-(t) \equiv 0$, they are

$$\begin{aligned}
E\big[\#P_1(t)\big] &= E\big[\#P_1^+(t)\big] = y_1 \ , \\
E\big[\#P_2(t)\big] &= E\big[\#P_2^+(t)\big] = y_2 \ , \\
\mathrm{Var}\big[\#P_1(t)\big] &= akt\mathrm{e}^{-kt} - ak^2t^2\mathrm{e}^{-2kt} \ , \\
\mathrm{Var}\big[\#P_1^+(t)\big] &= \begin{cases} a(4 - kt)\mathrm{e}^{-kt} - a(4 + 2kt + \\ \quad k^2t^2)\mathrm{e}^{-2kt} \ , & t \leq 1/k \ , \\ akt\mathrm{e}^{-kt} + a(2\mathrm{e} - 4 - 2kt - \\ \quad k^2t^2)\mathrm{e}^{-2kt} \ , & t > 1/k \ , \end{cases} \\
\mathrm{Var}\big[\#P_2(t)\big] &= \mathrm{Var}\big[\#P_2^+(t)\big] = a\mathrm{e}^{-k_2 t} - a\mathrm{e}^{-2k_2 t} \ .
\end{aligned}$$

$\mathrm{Var}\big[\#P_1(t)\big]$ and $\mathrm{Var}\big[\#P_1^+(t)\big]$ have been computed with $k_1 = k_2 = k$. In graphical form, they appear in Fig. 4. It is noteworthy that for all $t > 0$,



(a) GSPN model (b) SDA model with $k_1 = k_2 = k$

Figure 3: Partial state transition diagrams of a 2-phase radioactive decay process

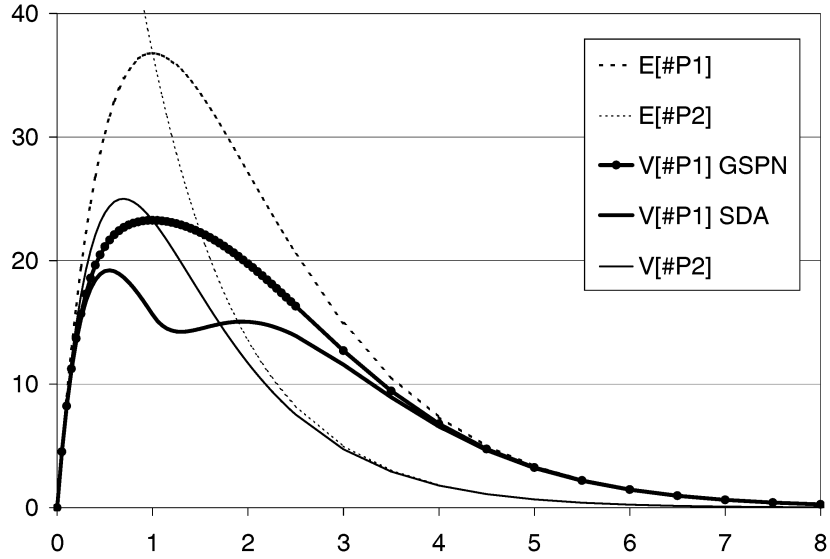
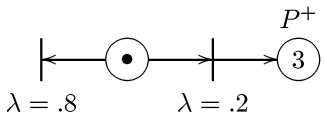


Figure 4: Expected values and variances vs. time with $a = 100,\ k_1 = k_2 = 1$

$\mathrm{Var}\big[\#P_1(t)\big] > \mathrm{Var}\big[\#P_1^+(t)\big]$, although the reachability set of the net in Fig. 2 is almost twice the size of the net in Fig. 1. Thus, an up to 20 % variance reduction is achievable with SDA.

INITIAL CONDITIONS

A single place marking cannot represent a non-integral initial value. Instead, a corresponding initial *distribution* has to be generated. For example, an initial value of 3.2 has to be represented with a distribution $\pi_3(0) = .8,\ \pi_4(0) = .2$. In GSPN terms, the net

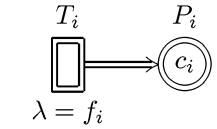


(7)

models the situation correctly.

META-SYNTAX

Admittedly, the syntax is crude and makes expressing large models tedious and graphically cumbersome. Therefore, we propose the following *meta-syntax*. Instead of using (2) and (7) to model (1), the following can be used.



$$f_i = a_{i1}\#P_1 + \cdots + a_{iN}\#P_N + b_i \tag{8}$$

In all marking expressions, $\#P_i$ denotes $(\#P_i^+ - \#P_i^-)$. While extending the meta-syntax of SDA to provide for

integration with the outside world is certainly desirable, it lies beyond the scope of this paper.

CONCLUDING REMARKS

We have introduced a method for simulating differential equations with GSPNs. A simple example showed that variance reductions compared to faithful modeling of the physical reality are possible. Nevertheless, variability of computed continuous measures is substantial, leading to potential difficulties when SDA components are interfaced to the outside world in a hybrid model. We are currently investigating advanced transient analysis algorithms that can avoid generating unwanted variance.

REFERENCES

Ajmone Marsan, O.; G. Balbo; G. Conte; S. Donatelli and G. Francheschinis. 1995. *Modelling with Generalized Stochastic Petri Nets.* John Wiley & Sons, New York.

David, R. and H. Alla. 1992. *Petri Nets & Grafcet.* Prentice Hall, UK.

Horton, G; V.G. Kulkarni; D.M. Nicol and K.S. Trivedi. 1998. "Fluid Stochastic Petri Nets: Theory, Application, and Solution Techniques". *European Journal of Operations Research*, 105(1), 184–201.

Karlin, S. and H.M. Taylor. 1984. *An Introduction to Stochastic Modeling.* Academic Press, Orlando, Fla.

Ojala, L.; O. Penttinen and H. Rantanen. 2003. "A Novel Application of Petri Nets: Simulation of Serial Quantum Computers—Feynman's Swap Computer". In *Proceedings of ICALP03 Satellite Workshop on Stochastic Petri Nets* (Eindhoven, the Netherlands). Universität Dortmund, Fachbereit Informatik, Forschungsbericht Nr. 780, 103–122.

Ojala, L. and O. Penttinen. 2003. "Simulating Quantum Interference in Feynman's $\sqrt{\text{NOT}}$-computer with Stochastic Petri Nets". In *Proceedings of ESMC03 Workshop Simulation with Petri Nets* (Naples, Italy).

PERFORMANCE MODELLING AND ANALYSIS

COUPLING FORMAL METHODS IN A PERFORMANCE MODELLING METHODOLOGY FOR ETHEROGENEOUS SUPERVISORY SYSTEMS

Francesco Moscato
Mauro Iacono
Nicola Mazzocca
Dip. di Ingegneria dell'Informazione
Seconda Università di Napoli, Aversa, Italy.
E-mail: (francesco.moscato,mauro.iacono,nicola.mazzocca)@unina2.it

KEYWORDS:
Multi-Formalism Modelling, Supervisory System, Real-Time Requirements, Petri Nets, Queuing Networks.

ABSTRACT

Real world systems are often heterogeneous, not only in their physical composition but also from a modelling and analysis perspective. Different components may be modelled by using the most suitable modelling formalism since no single formalism is adequate to model all system parts. In this work we present a modelling methodology that allows to use a multi-formalism methodology to model heterogeneous supervisory systems. The multi-formalism model is based on Generalized Stochastic Petri Nets (GSPN) and Queuing Networks (QN). The modelling methodology is applied in the design of a remote supervisory system of a robotized cell and the results are validated through comparisons with values measured on the real system after its implementation.

INTRODUCTION

Modelling and simulations are becoming increasingly important to help with the analysis and the design of real world complex systems whose components and aspects cannot be expressed by a single modelling formalism. To cope with the complexity of such systems multi-formalism approaches are emerging that allow using different formalism to model and analyze different subsystems and also to promote model reuse. Indeed multiformalism approach is very appealing, but proper methodology and technology must be developed.
In this work we introduce a methodology based on a multi-formalism approach to model heterogeneous distributed supervisory systems. Supervisory systems are generally employed to control and supervise industrial processes that, because of physical reasons depending on mechanical needs or human safety, may be distributed along space. The presence of sensors, continue and discrete controllers, Programmable Logic Controllers (PLC), general purpose PCs, interface cards, real-time and general purpose operating systems and different network infrastructures and protocols makes these systems heterogeneous in both hardware and software components. Since supervisory system tasks have to respect (soft and/or hard) real-time constraints, performance analysis and prediction is fundamental to design and tune the system to asses its normal working cycle. Timeliness, correctness of software components, sub-systems workload and bottlenecks have to be investigated also to assure stability in the plant control and to allow the system and the human supervisor to react to critical situations.
Petri Nets (PN) and Queuing Networks (QN) are widely used to estimate both hardware and software performances of complex systems (Raymond and Alain 1993, Ajmone Marsan et al. 1986). PN are used for qualitative (and in some evolutions also quantitative) analysis whereas QN are employed for performance evaluation (quantitative analysis). Only few efforts have been undertaken in the past to combine description features of QN and PN (Lin and Fu 1999, Bause 1993, Balbo et al. 1988). In particular, in (Lin and Fu 1999, Bause 1993) the original Generalized Stochastic Petri Nets (GSPN) formalism was extended by "hiding" queues in special places. This approach allows an analysis of the model of the whole system by using a single formalism in spite of an increase of complexity in the model.
In the following we present a methodology that allows to use a multi-formalism approach in modelling heterogeneous distributed supervisory system. The proposed methodology allows the performance analysis and the design of the whole system by dividing it in layers and subsystems, coupling the appropriate modelling formalism with each layer or subsystem. Furthermore an effective organization of the analysis process in layers helps the modeler to focus on the principal aspect each layer is involved in, and guides him in choosing the most appropriate modelling technique.

SYSTEM ARCHITECTURE

In this work we refer to a remote supervisory system architecture. According with (Miyagi et al. 2001) a *supervisory* system is defined inside a hierarchical structure as shown in Figure 1. The plant level is constituted by

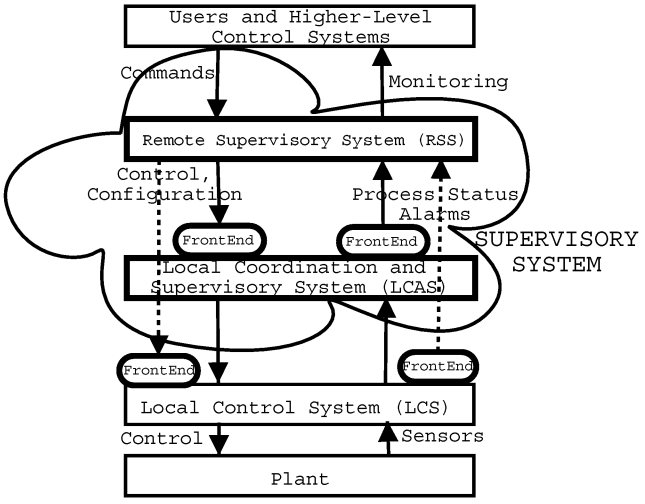


Figure 1. System Architecture

all non-computing physical devices. In the Local Control System layer (LCS), one or more local controllers interact directly with the plant: they could be discrete and continuous controllers, such as PLCs and PIDs. A Local Coordination And Supervisory (LCAS) layer is needed to perform the whole process since LCS may consists of more control units. Often, e.g. in manufacturing systems, a remote supervisory system (based on one or more general purpose PCs) is needed, for safety reasons. *FrontEnd*s allow the communication between local and remote systems, reading process variables from local plant and controllers and sending them to remote supervisors. These components are usually general purpose PCs, equipped with on-board special cards in order to interact with plant components and local controllers. Remote Supervisory Systems (RSS) are PCs connected to LCAS with common data networks (LAN, WAN). Users and Higher-Level Control Systems are employed for human interfaces and integration with other computing subsystems (e.g. for stock management or decision support).

We apply our methodology (explained in the next section) to LCS and RSS layers.

The main purpose of supervisory systems are :

- actions on LCS to perform tasks determined by users or any higher-level control system;
- supervision of the evolution of system dynamics and selection of the best local system configuration for global performance improvement;
- fault detection and diagnosis.

To achieve these goals the local and/or remote supervisor gets from the Local Control System process its status variables and eventually alarm notifications. These data must be elaborated under real-time constraints to supply monitoring information to the high-end users and to send automatically configuration or control signals to the local control system or to the plant. This is needed e.g. for fault tolerance or safety reasons or also to coordinate a distributed automation process (i.e. more robots, each locally connected to its own controller, sensors etc.). As a result, the complexity of the tasks reflects in a complex architecture, increasing the number of design parameters and complicating the performance prediction process. The nature of components and communications in the system is heterogeneous: different computing equipments are used in the system as seen, while control and configuration signals usually require different quality of services in term of real-time deadlines. In addition, data communication, process variable elaboration and eventually interface cards drivers and all other tasks on the supervisory system run concurrently on the computing nodes (general purpose PCs) with the aid of an operating system that manages real-time constraints.

THE MODELLING METHODOLOGY

The design process of a distributed supervisory system requires the modeler to capture some main issues in order to cope with the complexity of the system. The model is needed to tune hardware and software components in respect of real-time constraints associated to control and supervisory tasks. The main characteristics that the model of a supervisory system component must be able to capture are:

Concurrency and Parallelism: Tasks on FrontEnds and Supervisors take place simultaneously. *Conflicts* on same resources (such as shared memory variables) occurs if more tasks require the resource at same time. *Deadlock* or *Starvation* may occur if concurrency among tasks was not designed in the right way.

Real-Time Constraints: as explained in the previous section, all tasks in a supervisory systems must be executed under real-time constraints. This means that the elaboration of variables, the communication among distributed objects in the system and every other task in the system must finish its execution before the occurrence of a predefined deadline (determined at design time).

Queuing Effects: reliable communication protocols such as TCP use queues to manage incoming and outgoing packets or to implement some flow control mechanisms. Also the operating system scheduler uses queues to manage running and waiting processes.

Scheduling Policies: in a multi-programmed environment, operating system scheduling policies (FIFO, round-robin etc.) hardly influence response time (Tanenbaum 2001).

Communication: Communication hardware and protocols, especially non deterministic ones (such as Ethernet) must be considered at design time since they influence response time and correct behavior of each task.

To use the appropriate formalism for each (sub)system component, we characterize them in two levels : *task* level and *operating system and network* level. At task level, concurrency among tasks may cause excessive delays and/or deadlocks. At operating system and network level scheduling policies and scheduler characteristics (i.e. number of

scheduler priority queues, preemption, maximum number of active processes in the system etc.) may cause a wide variation on response time of each task. In addition, the communication protocol used (e.g. TCP/IP), the network infrastructure, the network load hardly affect response time mainly if the frequency of data transmissions is high and the number of tasks requesting them is large.

The proposed methodology is based on the employment of two different analytical modelling techniques, based on formal models: GSPN and QN. We choose to use analytical techniques rather than simulation techniques because of their intrinsic mathematical foundations which render the results general and the analysis independent from any specific tools (although computing can be affected by errors).

Due to the lack of space we do not describe the GSPN and QN formalisms; a more detailed discussion can be found in (Ajmone Marsan et al. 1986).

GSPN are used at task level to represent control tasks, supervisory tasks and interface drivers independently from the operating system scheduling effects. The formalism can be used to model and validate concurrent tasks and real-time constraints and also to find deadlocks while accessing shared resources (Balbo et al. 1988, Ajmone Marsan et al. 1995).

QN instead are used at operating system and network level to represent operating system schedulers and communication protocols behaviors.

Figure 2 shows how these two formalism are used to model supervisory systems.

At task level, GSPN models of control and supervisory tasks

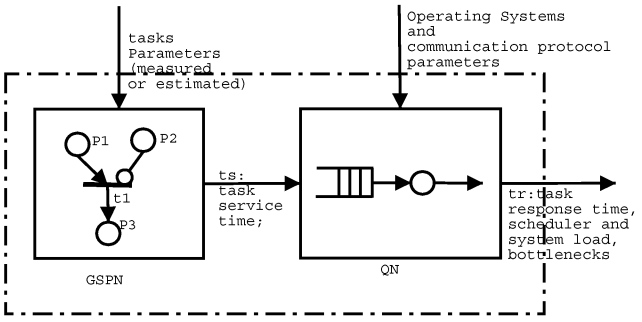


Figure 2. Modelling Methodology

and interface drivers are used to represent tasks behavior. Both concurrency among tasks and real-time constraints are modelled at this level but the tasks are considered as they were the only task active in the whole system. The outputs of this model are the execution times of each task (called *tasks service times* in Figure 2 because of their use as service times in the QN model described later). If tasks require data transmission, the GSPN model also provides, as model outputs, the transmission request distributions of each task. Since instructions or a set of them may be modelled by timed transition, it is necessary to evaluate the duration times of tasks instructions (or set of instructions). The measurement of these timings may be done *off-line* because all the effects of system actual load are considered in the next step of the methodology. Finally the GSPN tasks model can also verify (Ajmone Marsan et al. 1995) the presence of deadlocks if shared resource contests exist among tasks.

On the second hand a QN model is provided to model operating system and communication protocol when more tasks run concurrently or in parallel on the same computing node (FrontEnds and Supervisors in Figure 1). To model a scheduler behavior with a QN, the service time of scheduled tasks and their scheduling request distributions are required. These data are provided by the tasks GSPN model. In the same way the queuing effects of transmission protocol such as TCP/IP are modelled. Since the formalism used at operating system level and communication protocol level is the same (QN formalism), a unique model can be used at this level. The outputs of the QN model are the effective execution time of each task (*response time* in terms of QN) considering the real load of the system. The QN model also characterizes the system bottlenecks evaluating scheduler and network load. The results from the model may be used to tune the system to respect real-time constraints. It is possible to evaluate the possibility of increasing or decreasing the number of supervisory tasks or the frequency of transmission operations needed, for example, to read plant process variables from remote supervisors.

CASE STUDY

Overall system description

The case study application is a remote Supervisory Control And Data Acquisition (SCADA) system for a robotic cell which performs a palletizing process.

The robotic cell consists of two COMAU robots and their control units, a conveyor belt and an Allen-Bradley PLC (SL5/04) together with othe minor devices. The COMAU robots handle the pieces by means of hydraulic clamps on their extremities: the first robot gets the pallets from a tray and puts them on a conveyor belt that transports the pieces toward the second robot; the second robot gets the pallets and arrange them on the second tray. The two robot control units transmit, if requested, the state of the robotic arms in terms of (radial) position of the axes, beans thermal state, singularity point positions, velocity and acceleration values for each axes and so forth. The PLC locally supervises the process, by using a predefined sequence of instructions to perform the palletizing process. It is also possible to reconfigure the process by sending control and configuration commands to the PLC. Figure 3 shows the configuration of the SCADA system of the described robotic cell installed at the University of Naples. It consists of two general purpose PCs: the first has a module interface *Interface* towards the PLC on-board and act as FrontEnd; the second one works as remote SCADA system: it demands to the local FrontEnd the interaction with the PLC. The remote SCADA communicates with the FrontEnd through an Ethernet link.

To supervise the palletizing process the SCADA system and the FrontEnd exchange control, status and configuration data.

The FrontEnd interface card driver reads and writes variables from and to the PLC memory using a polling protocol. The variables are stored in the FrontEnd shared memory accessed by the PLC driver and other FrontEnd processes.

Remote SCADA reads from FrontEnd the process status

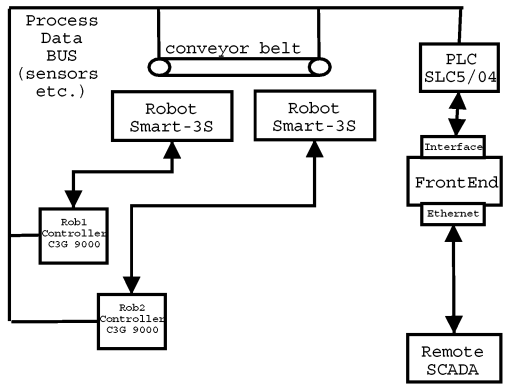


Figure 3. SCADA Architecture

(i.e.: robot axes positions, conveyor belt direction, optical and inductive sensors state, etc.) and the alarm variables that reveal the occurrence of one or more faults in the process, in order to produce a representation of the palletizing process through a graphical user interface and to automatically perform some control actions depending on process status. All these operations are demanded to different threads. The read variables are stored in the SCADA shared memory to allow the other threads to easily and quickly access the needed information. All threads executed on the FrontEnd and on the remote SCADA are scheduled under (soft) real-time constraints. A model of the whole system, aimed to design and tune the remote SCADA, system was built following the methodology described in the previous section. The goals of our analysis are:
- to locate system bottlenecks;
- to provide parameters to be used in system tuning phase in order to guarantee that the required real-time deadlines will be met. In particular we want to evaluate the response time (t_r) to all SCADA requests towards FrontEnd to tune:
a) the number of control and configuration threads both on FrontEnd and remote SCADA;
b) the width of the polling interval among two consecutive thread requests.

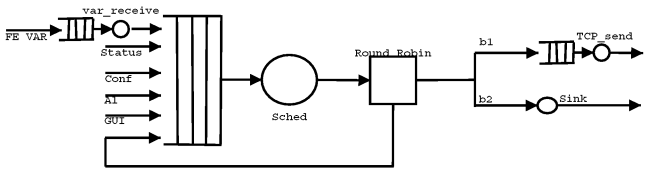


Figure 4. SCADA QN model

SCADA QN Model

The response time to SCADA requests t_r and the informations to evaluate system bottleneck can be retrieved as output of the QN model of the SCADA system reported in Figure 4 where *Al*, *Status* and *Conf* represent respectively the requests sent to FrontEnd to :
a) get process alarm variables;
b) get process status variables;
c) perform control and management operation on the process.
GUI, instead, represents the periodical GUI upgrade process and FE_VAR represent the arrival of process variables from the FrontEnd. Some threads need to send process variables requests to the FrontEnd (b_1), while other tasks do not ask for further operations (b_2). The scheduling policy used by operating system is a preemptive round-robin (represented in Figure 4 by the rectangle at the left of server *Sched*). Communication-related queues (*Req and TCPsend*) are modelled by a **M/M/1** queue. A time slice q is associated to the round-robin preemptive scheduler queue. Assuming in the following :
- n_i*q+h_i (h<q) is the time units used by thread i to accomplish its task;
- t_0 is the service time of scheduled activities requiring a whole time slice (all (sub)task preempted);
- $t_{s_i}, i \in 1, ..ntask$ the service time of scheduled activities requiring less than a time slice to terminate;
- p_i , $i \in 0, ..ntask$ the relative frequency of each (sub)task.
It is possible to prove (Moscato 2002) that our model can be studied as an **M/G/1** queue with :

$$t_s = \sum_{i>=0} t_{s_i} * p_i; \qquad \sigma_s^2 = \sum_{i>=0} (t_{s_i} - t_s) p_i; \qquad C_s^2 = \frac{\sigma_s^2}{t_s^2}$$

The proof is omitted due to the lack of space.

SCADA GSPN Model

To resolve the QN model the arrival distributions and the service times of *Al*, *Status*, *Conf*, *GUI* and FE_VAR, requests must be provided. While arrival distribution and service time of *GUI* requests is fixed at design time, *Al*, *Status*, *Conf* service times are the outputs of the GSPN models of tasks running on the SCADA System. In addiction arrival distributions and service time of FE_VAR requests are outputs of FrontEnd QN model of Figure 6 described later in this section. In Figure 5 the model of two concurrent thread

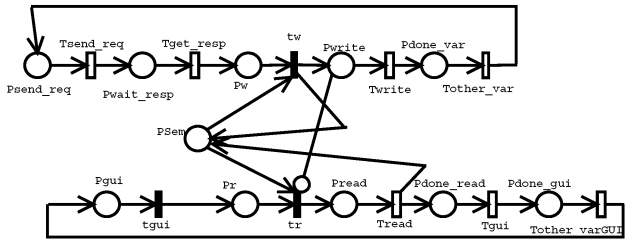


Figure 5. GSPN model of two SCADA threads

on the remote SCADA is reported . The first thread (on the top of the figure) sends requests to get the process status variables from the FrontEnd and stores them in the remote SCADA shared memory. The second thread (on the bottom of the figure) reads from this shared memory and upgrades the graphical representation of the pallettizing process on the remote SCADA Graphical User Interface (GUI).

The throughput of the timed transition $T_{sendReq}$ represents the distribution of *Status* requests arrival times of model in Figure 4 and the throughput of the transition t_{GUI} represents the distribution of *GUI* requests arrival times of the same model. The model also provides the mean service time of a request or a GUI operation without considering scheduling or communication protocols effects. These times are used as service times in the QN SCADA model.

FrontEnd QN Model

In the same way a model of FrontEnd system is realized to get information about FE_VAR requests. The QN

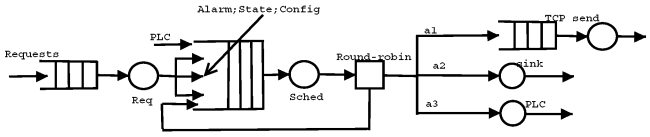


Figure 6. FrontEnd QN model

model of the FrontEnd is reported in Figure 6. The remote SCADA sends requests to the FrontEnd to : a) get process alarm variables; b) get process status variables; c) perform control and management operation on the process. These three kind of requests are managed by different threads that are scheduled by the operating system together with the PLC interface card driver. The scheduling policy used by operating system is preemptive round-robin (represented in Figure 6 by the rectangle at the left of server *Sched*). To accomplish their tasks, some of the scheduled threads require to write on PLC memory or send feedback informations to the remote SCADA. In Figure 6 a_k $k \in \{1, 2, 3\}$ represents the percentage of scheduled operation requiring: (a_1) send operations towards remote SCADA; (a_2) tasks that do not ask for further operations; (a_3) PLC operations.
Alarm, State and *Config* request arrival distributions in Figure 6 are represented by the outputs of the QN SCADA model. The frequency of PLC operations is instead a design parameter. The requests service times are the outputs of the GSPN FrontEnd tasks model.

FrontEnd GSPN Model

Due to the lack of space in this section only the model of two concurrent tasks of the FrontEnd is reported (Figure 7). The first task performs the send and receive operations towards the remote SCADA system and interacts with the second task (PLC interface card driver) through read and write operations on the FrontEnd shared memory. On the left of Figure7 the protocol used by the first task to access to the shared memory is modelled. The middle part of the model in Figure7 describes read-write contests in a shared memory managed by a semaphore (P_{sem}). Transitions T_read_PLC and T_write_PLC represent read and write operations from and to the PLC memory respectively. Transitions T_prep_pkt and T_{send_pkt} represent the operations of sending data and feedback information to the remote SCADA.
Notice that read-write conflicts affects the execution time

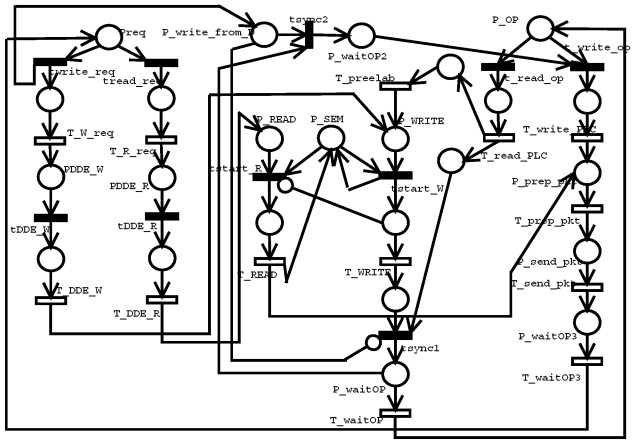


Figure 7. GSPN model of two FrontEnd threads

of each thread. In the following we report (resumed in the Table 1) some results obtained by solving the multi-formalism model explained in the previous paragraph. The

Table 1. Some Experimental Results

Parameter	*Model*	*Value measured on real system*	*Variation*
$t_{sAL,sST,sC}$	3.27msecs	3.39msecs	3.8%
t_{sGUI}	20.03msecs	20.83msecs	4.0%
t_r	15.89msecs	16.62msecs	4.6%
Deadline expiration Probability(t_r)	0.004	—	—

values of $t_{sAL,sST,sC}$ represents the mean values of services times for the scheduler *Sched* in Figure 4 of the remote SCADA (Alarm, State and Config) requests. These values are the outputs of the GSPN submodel reported in Figure 7. Likewise $t_{sAL,sST,sC}$, t_{sGUI} represents the mean value of service times for GUI operations for remote SCADA and is obtained by solving the model in Figure 5. Finally t_r represents the mean value of the response time to requests sent by the remote SCADA to the active FrontEnd to read process status variables. These requests are scheduled under (soft)real-time constraints: the deadline for a request operation is set at 30 msecs. The probability of a deadline expiration for a request operation is evaluated on the basis of the distribution of the requests response time obtained by solving the submodel in Figure 4. The results obtained by the multi-formalism model have been verified against the real values measured on the supervisory system after its implementation. Table 1 includes some results from both cases and their percentage variations. The variations are less than 5 % for each parameter.

CONCLUSIONS

The presented modelling methodology, applied to the case study of a remote SCADA system, efficiently and effectively allows to define and use predictive performance

models by composing them from heterogeneous submodels to build a model of a complex supervisory system. The proposed methodology allows to use different formalism for different problems in different levels of modelling and provides a modular approach to system tuning, supporting reuse of submodels. As shown in the *Case Study* section, (sub)models are simple and are solved independently with advantages for the evaluation phase.
Future works include the integration of the technique in a multi-formalism modelling framework in order to automate the solution process and to evaluate the opportunity of introduction of other kind of models (including also simulative models) in the methodology itself and to define all the composition rules to perform their integration.

REFERENCES

Ajmone Marsan M., Balbo G. and Conte G. 1986. *Performance models of multiprocessor systems*. MIT Press, London.

Ajmone Marsan M., Balbo G., Conte G., Donatelli S. and Franceschinis G. 1995. *Modelling with Generalized Stocastic Petri Nets*. Wiley Series in Parallel Computing.

Balbo G., Bruell S.C. and Ghanta S. 1988. "Combining Queueing Networks and Generalized Stochastic Petri Nets for the solution of complex models of system behavior". In *IEEE Transaction on Computers, Vol.37*, No. 10 (Oct.) 1251-1268.

Bause F. 1993. "Queueing Petri Nets - A Formalism for the Combined Qualitative and Quantitative Analysis of Systems". In *Proc. 5th International Workshop on Petri Nets and Performance Models* (19-22 Oct 1993), IEEE, 14-23

Lin M.-H. and Fu L.-C. 1999. "Modeling of Priority Queueing Service in discrete Event System Using Hybrid Petri Nets". In *Proc. IEEE International Conference on Systems, Man and Cybernetics* Vol.1.

Miyagi, P.E., Villani E. and Maruyama N. 2001. "Modeling of Hybrid Supervisory Systems using UML and Petri Nest. Proceedings". In *8th International Conference on Emerging Technologies and Factory Automation*, Vol.2, 149-158

Moscato, F. 2002. "Realizzazione e validazione di un sistema remoto per il controllo di una cella robotizzata" (in italian). *Master's thesis*, Universita' di Napoli "Federico II", Naples, Italy.

Raymond M. and Alain J.-M. 1993: "Quantitative Evaluation of Discrete Event Systems: Models, Performances and Technique"s. In *Proc. 5th International Workshop on Petri Nets and Performance Models* (19-22 Oct 1993), IEEE, 2-11.

Tanenbaum A. S. 2001. *Modern Operating System*. Prentice Hall.

APPROXIMATE AND SIMULATION BASED ANALYSIS FOR DISTRIBUTED OBJECT SOFTWARE PERFORMANCE MODELS

Panajotis Katsaros and Constantine Lazos
Department of Informatics
Aristotle University of Thessaloniki
54006 Thessaloniki
Greece
E-mail: {katsaros,clazos}@csd.auth.gr

KEYWORDS
synchronous object invocation, nested object invocations, simultaneous resource possession, approximate MVA, flow-equivalent approximation, regenerative method

ABSTRACT

In complex software systems, the effectiveness of model based performance predictions is limited by the availability of appropriate solution techniques. These techniques should allow to take into account the software components interaction effects. In distributed object systems, the main problem is the simultaneous resource possession caused by the synchronous, often nested object invocations, which block the callers, until they get the replies. This paper provides a review of the analysis techniques, which address that fact, while preserving the abstract system view, offered by a queuing network representation. Two of these techniques, were proposed for solving a general class of models, with one or more layers of software servers and a third technique was designed specifically for distributed object software performance models. The advent of an extended flow-equivalent approximation, which is also described, opens new prospects for the development of efficient solution algorithms. Finally, simulation based estimation is discussed, in respect with the applicability of the well-founded and accurate, single-run regenerative method.

1. INTRODUCTION

Classical queuing network solution techniques, assume that a job can only use a single resource at a time and simultaneous resource possession is not possible. Moreover, most of them may be only used for analyzing typical "flat" system representations and do not directly support the solution of excessively complex queuing networks, like the ones obtained in realistic performance models of distributed software systems.
Early contributions to the problem of analyzing simultaneous resource possession, were based either on

- iterative estimation techniques with unknown calculation time and not guaranteed convergence for the performance measures of interest or
- low-level Markov based approximations that are neither practical nor scalable enough for software engineering purposes.

Recent advances, like the advent of an extended flow-equivalent approximation (Kurasugi and Kino 1999), constitute a new theoretical basis for the development of efficient solution algorithms.
In this paper, we review those techniques that we believe, they may have an impact in the practice of distributed object systems performance modeling and we comment on the new prospects opened by the last developments.
Accurate simulation based analysis of such complex models is an absolute necessity, since it is basically the only alternative for checking the validity of the approximations to be used. Thus, we also review theoretical results on the applicability of the single-run regenerative approach, in hierarchical queuing network models.

2. ITERATIVE PERFORMANCE EVALUATION ALGORITHMS

The Stochastic Rendezvous Networks

The Stochastic Rendezvous Networks (SRVNs) were first introduced in (Woodside 1989) and constitute a practical and scalable technique for the analysis of systems, with software components interactions. The rendezvous mechanism can be used to model different interaction types present in software, including all the types of object invocations (Woodside et al. 1995) taking place in distributed object systems. The solution for a SRVN can be found with an iterative algorithm that computes a series of intermediate solutions, using an MVA approximation and continues until the estimated throughputs converge.
The primitive structural entity of a SRVN model is the *task*. Each task may represent a single software or hardware component that communicates with other tasks, by messages of a request-wait-reply style. The task, which sends the message (*client task*), requests the rendezvous and blocks until it gets the reply. The receiving task (*server*) accepts the request and executes two (or more) service phases. The client task is blocked only for the time period of the first phase. Thus, a synchronous object invocation may be modeled by a rendezvous request with zero service requirements, for its second phase. On the contrary, asynchronous (one-way) object invocations are modeled as rendezvous requests with zero service requirements, for their first phase.
A SRVN differs from a typical queuing network, in that a server may also act as a client requesting service from one

or more lower layer tasks, in any phase of a rendezvous service execution. Each server may provide more than one service, modeled as separate task *entries*, with their own parameters. However, each task owns a single message queue with a specific service discipline. Entries correspond to distinguished object method invocations, since messages are directly addressed to the entries and not to the tasks.
There are three types of tasks, namely:

- the *pure clients*, labeled as 1, 2, . . . , R, which may only initiate requests,
- the *active servers*, labeled from R+1 to K, which may accept requests and initiate new ones and
- the *pure servers*, labeled from K+1 to N, which may only provide service to accepted requests.

Processors are pure server pseudo-tasks that provide hardware resource service to one or more active server tasks. The entries of the tasks, assigned to them, request service one slice at a time and each request and service are modeled in the SRVN as a rendezvous between the requesting task and the processor pseudo-task.
The SRVN solution algorithm, proposed in (Woodside et al. 1995), is summarized in the following steps:

1. Carry out the software model transformation to create tasks that represent processors, for co-allocated tasks. This involves:
   a. define a new pseudo-task entry ε for each entry e of all the tasks co-allocated at the same processor and an arc from e to ε for its processing requests
   b. label the arc with the mean number of processing slices, in phase p, calculated as
   $$y_{e\varepsilon p} = 1 + \sum_{d \in \varepsilon}^{*} y_{edp}$$
   where y_{edp} are the parameters denoting the mean number of entry e requests to d, during phase p
   c. if s_{ep} is the input denoting the mean service time provided by entry e, during phase p, set
   $$s_{\varepsilon 1} = \frac{s_{ep}}{1 + \sum_{d \in \varepsilon}^{*} y_{edp}}$$
   and s_{ep}=0, since the processor pseudo-task is a pure server and each entry ε of it owns only phase 1
2. Construct the so-called *task request graph*, with a node for each task and a directed arc from i to j, if the entry e in task i sends requests to any entry d in task j. This graph is used to determine the order of computation in the next steps. If there is a cycle in the task request graph, it is not possible to determine such an ordering.
3. Let us denote:
   λ_e, the unknown throughput of entry e in messages/sec
   λ_{0e}, the parameter specifying the arrival rate, if there is an external stream of requests to entry e
   $Y_{de} = \sum_{p} y_{dep}$, the mean number of entry d requests to entry e, during all phases p

   Then, $\lambda_e = \lambda_{0e} + \sum_{d \in \varepsilon}^{*} Y_{de} \cdot \lambda_d$, for all e from R+1 to N
   and this constitutes a set of N - R equations, with N unknowns. This may be also written as
   $$\lambda_e = \sum_{r=1}^{R} \alpha_{er} \cdot \lambda_r + \sum_{d=R+1}^{N} \beta_{ed} \cdot \lambda_{0d} \qquad (1)$$
   for all e from R+1 to N, where the coefficients α and β are determined by applying Gaussian elimination to the original set of equations. Set w_{kl} = 0, with w_{kl} denoting the mean waiting time seen by entry k requests, when sending messages to entry l, for all k, l with Y_{kl}>0.
4. Calculate mean service times (including the queuing delays) and throughputs, in the order found at step 2: For each task i, the mean service time is given as
   $$x_{ep} = s_{ep} + \sum_{d \in \varepsilon}^{*} y_{edp} \cdot (t_{ed} + \delta_{ed})$$
   with t_{ed} the mean delay for entry e, when sending a message to entry d and δ_{ed} is the specified mean round-trip communication delay from entry e to d.
   For FIFO tasks, $t_{ed} = w_{ed} + x_{b1}$, since entry's e task remains blocked only for the duration of the first phase of the rendezvous request. Throughputs are given as
   $$\lambda_r = \frac{1}{\sum_{p} x_{rp}}$$
   for all entries r from 1 to R and from (1), for all entries from R+1 to N.
5. Estimate the mean waiting times for all entry requests directed to other entries, based on the MVA expression provided in (Woodside et al. 1995) and the alternative approximations proposed, in order to determine the task queue properties, at the instant of the rendezvous request arrival.
6. If throughputs are all sufficiently close to the previous iteration, then stop. Otherwise go to step 4.

The results obtained include throughputs, mean waiting times, entry service times and entry and task utilizations. The described solution technique is closely related to the Bard-Schweitzer approximate MVA (Schweitzer 1979) and the heuristics provided in (Woodside et al. 1995) support either FIFO or Preemptive Priority scheduling and stochastic or deterministic phase types, with possibly varied coefficients of variation, in the second case.
The SRVN based solution variants cannot be applied to models with multi-threading tasks.

The Method of Layers

In the method of layers (Rolia and Sevcik 1995), the system performance model is viewed as a sequence of layers, where processes request service only from one level lower, in the hierarchy. Processes with statistically identical behavior form a *group* or a class of processes. To identify the level of each group, in the hierarchy, a topological sort of the process groups may be first developed. Thus, each group is placed at exactly one level and the resulted graph is assumed not to contain any

cycles. If requests for service span more than one level, virtual flow-equivalent groups modeling the requests and their service rates have to be introduced, in all intermediate layers (this is not required for the SRVNs based solution techniques).

Each pair of successive levels, in the hierarchy, defines a submodel. The response time of a process that is considered as a client in one submodel defines its service time, when it is considered as a server in its alternate submodel. The performance estimates for the submodels are found using a modified version of the Linearizer algorithm (Chandy and Neuse 1982). The solution of all of the submodels represents an iteration of the algorithm. The algorithm is applied iteratively, until the changes in estimated mean response times, between successive iterations, is below a specified tolerance.

When solving the hierarchy of models only software contention is taken into account. After the algorithm terminates, a second model, in which software contention is ignored, is created. Each process is included in a single hardware resource contention queuing network and the amount of time that a process does not spend competing for devices is used as a think time in the model.

With new estimates of device responsiveness, another solution for the software contention model is found. The method of layers alternates between software and hardware contention models, until the estimates for mean response times of non-serving processes differ by less than some tolerance. This approach is similar to the method of complementary delays (Heidelberger and Trivedi 1983) that proceeds by using closed queuing networks, augmented by a virtual delay service center.

The method of layers supports the use of multiple entry FIFO rendezvous servers, as in the case of the SRVNs based solution techniques. Additionally, the use of the so called multi-servers permit the representation of multiple thread servers (as opposed to the SRVNs based solution techniques).

The combination of the use of the Linearizer, which often provides superior accuracy, when compared to the Bard-Schweitzer algorithm and the simultaneous solution of all servers in a layer yields, in some cases, more accurate results, compared to the ones obtained by the SRVN based solution variants. In the sequel, the basic method of layers estimation is outlined:

1. Initialize the response time estimates for process groups, with the mean service times, assuming no hardware or software contention. The initial response times are computed level by level in a bottom-up manner.
2. WHILE group response times have not reached a fixed point DO
   WHILE group response times have not reached a fixed point DO
   FOR each software submodel selected in a top-down manner DO
   solve the submodel using Linearizer with the following residence time expressions (Rolia and Sevcik 1995): FIFO, rendezvous, multiple entry, multi-server and delay
   END FOR
   END WHILE
   solve the hardware resource contention submodels using Linearizer with the following residence time expressions (Rolia and Sevcik 1995): PS, FIFO, LIFO, priority preemptive resume, and delay
   Update the group response time estimates
   END WHILE

The Method of Decomposition

The method of decomposition, introduced in (Kahkipuro 2000), was designed specifically for distributed object software performance models. It includes approximations, which overcome the limitation of not being applicable (as in the case of the methods already described) in models with cyclic graph dependencies. Cycles may be introduced, when two objects are allowed to make synchronous invocations to each other, as for example in the case of the callback interaction pattern. This type of interaction is sometimes used as a means to avoid extensive blocking times that may be caused by the use of a single synchronous object invocation. Recursive invocations to the same object are allowed, since this is also possible in real world applications.

In the sequel, we provide a description of the core solution technique. The overall approach lies on the well-known method of surrogate delays (Jacobson and Lazowska 1982), in order to decompose the model into multiple queuing networks, so that the primary and the secondary (software or hardware) resources of a blocking access are always in different networks. As a result, we obtain a set of multi-chain product form queuing networks.

For each case of simultaneous resource possession, the primary network is created by removing all secondary resources and all accesses to them and then by including a surrogate delay resource. The following transformations are carried out:

- for each access to the primary resource, its service demand is increased by the sum of the response times obtained for all blocking invocations to the secondary resources
- for each chain (class), the service demand of the surrogate delay server is defined to be the sum of the response times for all non blocked accesses of the secondary resources used by the chain

For each primary network, a set of secondary networks is created. These networks contain the secondary resource itself and an auxiliary delay server, for modeling the time the jobs spend elsewhere in the system. Each chain in the original queuing network generates at most two secondary chains:

- a closed one, if there are blocking accesses, with its population defined as the maximum number of accesses that can reach the secondary resource in parallel (multithreading server) and
- a secondary chain that corresponds to a chain with non blocking accesses to the resource, with the same parameters (equal arrival rates in case of open chains and equal populations in case of closed ones)

The service demands for all non blocking and blocking secondary resource accesses remain the same, unless there are nested blocking accesses to other resources. In this case, the service demand is increased by the sum of the response times obtained for the nested accesses. For closed secondary chains the service demand imposed to the auxiliary delay server is calculated through the Little' s rule, by using the throughput of the calling resource, in the corresponding primary network. For open secondary chains there is no need to specify a service demand for the auxiliary delay server.
The resulted networks are to be solved by an approximate Bard-Schweitzer MVA algorithm, modified appropriately (by the load concealment transformation, as described in Agrawal 1985) for the simultaneous solution of the open and the closed chains. Input parameters for some networks require the existence of a solution for some other networks and it is even possible to obtain cyclic dependencies. Thus, the solution is carried out iteratively and in each repetition the input parameters of the networks are adjusted to the outcome of the previous repetition, until the throughputs of the secondary resources are sufficiently close to the throughputs of the corresponding primary ones.
To conclude, the method of decomposition does not support priority scheduling, as opposed to the SRVNs based solution, proposed in (Woodside et al. 1995) and the method of layers.

3. AN EXTENDED FLOW-EQUIVALENT APPROXIMATION

The standard decomposition or flow-equivalent method (Courtois 1977 and Chandy et al. 1975) is a well known approximation, based on the replacement of a subset of the model's queuing centers, with a single, flow-equivalent center, characterized by a calculated state-dependent service rate.
Flow-equivalent based performance estimation techniques have been used in various simultaneous resource possession models (Sauer 1981). However, these methods could not be applied to models, where jobs that possess different types of passive resources (i.e. different sets of blocked object servers, as in the case of synchronous nested object invocations) compete with other jobs for the same active resources.
Recently, in (Kurasugi and Kino 1999), the authors introduced an extended flow-equivalent approximation, for overcoming this particular restriction. The proposed approximation is described in the frame of a general two-layer queuing network, where the upper layer represents software resource contention and the lower layer, the caused hardware resource contention (Figure 1).
The existence of a single upper layer does not provide adequate support for the representation of nested object invocations, where multiple object servers may be blocked at the same time. However, the advent of such an extended flow-equivalent approximation opens new prospects for the development of efficient non-iterative estimation techniques and for this reason, we proceed to the description of it.

Let us consider a queuing model, where the upper layer consists of R FIFO software servers labeled as 1, 2, . . . , R and the lower layer consists of N queues representing hardware resources that are labeled as 1, 2, . . . , N. Each software server owns k_i ($\geq$1) threads, $i \in 1, \ldots, R$ and one queue.

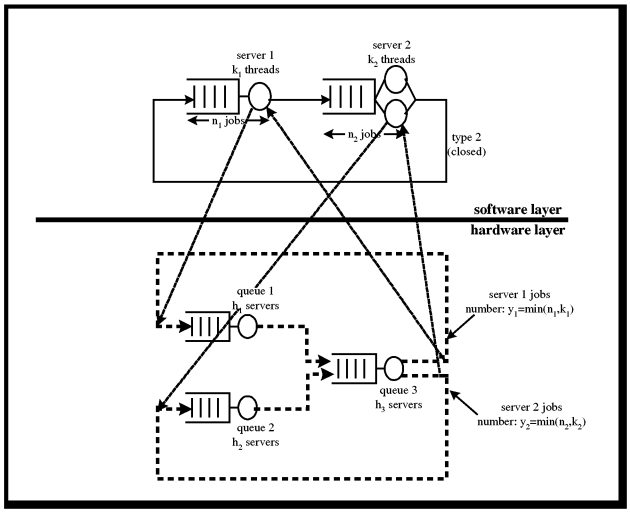


Figure 1: A two-layer queuing network with simultaneous resource possession

A job (object invocation) that arrives at server i will immediately be routed to the lower layer upon its arrival, if there is an available thread at the server. The time period in which a job passes to and returns from the lower layer is referred to as the server's service time. Each queue $j \in 1, \ldots, N$ in the lower layer owns h_j ($\geq$1) servers and one queue. Each job routed to the lower layer from software server i will travel through the queues at that layer, according to a specified Markovian routing chain, associated with the software server i. Subsequent routing of a job (object invocation) through the various software servers, in the upper layer, can be conducted according to a number of possible (open or closed) routing chains, that represent the desired software functionality. In this article, our description, for the upper layer, is restricted to the simple case of a single closed routing chain, with v_i denoting the relative frequency for a job to visit the software server i.
Let $x_{j(i)}$ be the total number of jobs of software server i, placed at the lower layer queue, j. The number of jobs of software server i, in the lower layer is given as,

$$y_i = \sum_{j=1}^{N} x_{j(i)}$$

and the aggregate queue occupancy vector is defined as

$$\mathbf{y} = (y_1, y_2, \ldots, y_R)$$

If n_i is the total number of jobs (object invocations) at the software server i, then y_i=min(k_i,n_i).
For any job arriving at the queue j, the required service time will depend on the particular software server the job comes from and is subject to an exponential distribution with parameter $\mu_{j(i)}$. The service discipline at each lower layer queue can either be FIFO, PS (Processor Sharing) or LIFOPR (LIFO Preemptive Resume). If the discipline is

FIFO, the parameters $\mu_{j(i)}$ are the same for any i, i.e. $\mu_{j(i)} = \mu_{j(h)}$, for any $h \neq i$.

The extended flow-equivalent method, proposed in (Kurasugi and Kino 1999), is summarized in the following steps:

1. Calculate the numbers of jobs $\mathbf{y} = (\min(n_1, k_1), ..., \min(n_R, k_R))$ in the lower layer, for all possible software server vectors, $\mathbf{n} = (n_1, n_2, \ldots, n_R)$.
2. Each **y**-job based model is created from the original two-layer model, by disregarding the behavior of jobs in the upper layer and setting a short circuit at each software server, so that each job arriving at the server from the lower layer returns immediately to it. Thus, each short-circuited server can be effectively replaced by a state-dependent virtual service center, whose service rate is calculated in the following manner:
   a. Solve the shaped **y**-job based models, for all **y**, by using the convolution algorithm, in order to obtain the normalization constants G(**y**).
   b. Approximate the service rate $\mu_i^*(\mathbf{n})$ at the software server i as
   $$\mu_i^*(\mathbf{n}) = v_i \cdot \frac{G(\mathbf{y} - \mathbf{e}_i)}{G(\mathbf{y})}$$
   where $\mathbf{e}_i$ = (0, . . ., 0, 1, 0, . . ., 0), with the unit placed in the i-th position.

In (Kurasugi and Kino 1999), the authors also propose a Markov based approximation and an alternative closed form approximation, for the upper layer, which we think are inappropriate for the performance evaluation of nested object invocations. Thus, we omit their description.

The extended flow-equivalent approximation of (Kurasugi and Kino 1999) may be used as is, in the hybrid simulation setting, in the way described in the next section. On the other side, we continue working to take advance of it, in order to provide a novel estimation technique that will successfully address the needs for efficient performance prediction, under the existence of synchronous, nested object invocations, in the software layer. The software servers' workload derivation procedure, to be used, will be based on the one introduced in (Vetland et al. 1993).

4. SINGLE-RUN SIMULATION BASED PERFORMANCE EVALUATION

Recently, (Nikolaidou and Anagnostopoulos 2003) introduced a network/application oriented modeling approach, for distributed system simulation. In (Katsaros and Lazos 2002) we focued on the development of (hybrid) simulation queuing models, for object based distributed software. Alternatively, (Brosso et al. 2002) introduces the use of XML Petri Nets (PNML), as a means for structured Petri-net model development. Also, recent advances (Anagnostopoulos and Nikolaidou 2003) in Faster-than-Real-Time simulation opened new prospects in model to system validation. In this section, we are restricted to the single-run output analysis of complex hierarchical model configurations, like the ones described.

We propose the use of the so-called regenerative method for sequentially controlled estimation of steady-state performance measures, for the following reasons:

- it is based on a sound theoretical foundation,
- it is not bound to the problem of the "system initialization bias",
- it is characterized by appealing asymptotic properties,
- its accuracy has been tested both in "flat" (Katsaros and Lazos 2003a), as well as, in hierarchical queuing network models (Blum et al. 1985) and
- it is also applicable in the hybrid simulation setting, with state-dependent service rates (Schedler 1993).

Although this method is not in widespread use, due to an inherent difficulty in identifying regeneration system states, there is significant theoretical progress towards this direction, in the last few years. The most notable advances rely on a generalized semi-Markov (GSMP) system representation and conclude to general results that may be utilized, for the regeneration state selection, at the more abstract level of a queuing network model (Schedler 1993). In (Katsaros and Lazos 2003a), we have experimentally evaluated the effectiveness of alternative selection strategies, in the sequentially controlled setting, in terms of both the obtained accuracy and the observed efficiency behavior.

5. CONCLUSIONS AND FURTHER RESEARCH WORK

This work is focused on the available approximate solution methods that we believe they may have an impact in the practice of distributed object systems performance modeling. They are all based on iterative estimations that scale well in large software performance models, but are also characterized by unknown calculation time and not guaranteed convergence, for the measures of interest.

The SRVNs based solution techniques lack support for cyclic object invocation dependencies, recursion and multithreading servers, since they were initially designed for the performance analysis of general software systems. The method of layers allows the use of multithreading servers, but requires that processes request service only from one level lower, in the hierarchy. Both approaches offer support for synchronous, as well as asynchronous object method invocations, by the use of the two-phase rendezvous type interaction mechanism and multiple entries software servers.

Priority scheduling is mainly required for the modeling of real-time distributed object systems (Katsaros and Lazos 2003b) and is also included as an open possibility for the two aforementioned methods. On the other side, it is not supported by the method of decomposition, which, as opposed to the two other alternatives, allows cyclic object invocation dependencies (callbacks) and recursion.

Although there are still great challenges for improvements in the existing core estimation techniques and the specialized approximation heuristics used by them, the advent of the extended flow-equivalent approximation,

described in section 3, opens new prospects for the development of new, efficient solution approaches.
Significant progress has been also carried out in the field of model parameter capture (Chatzigeorgiou 2003 and Vetland et al. 1993).
Simulation based analysis for hierarchical model structures like the ones used in the described analysis methods is an absolute necessity, since it is basically the only alternative for checking the validity of the used approximations. The proposed single-run output analysis method is based on a sound theoretical foundation, it is not bound to the problem of the "system initialization bias" and it is also characterized by appealing asymptotic properties. Furthermore, its accuracy has been tested in "flat" (Katsaros and Lazos 2003a), as well as, in hierarchical queuing network models (Blum et al. 1985) and we have also theoretically checked its applicability in the hybrid simulation setting, with state-dependent service rates.

REFERENCES

Agrawal, S. C. 1985. *Metamodeling: A study of approximations in queuing models*. MIT Press, Cambridge, MA, USA.

Anagnostopoulos, D. and M. Nikolaidou. 2003. "Timing Issues and Experiment Scheduling in Faster-than-Real-Time Simulation". (To appear in) *Transactions of the Society for Modeling and Simulation*. SCS.

Brosso, I.; G. Bressan and W. V. Ruggiero. 2002. "Simulation e-business applications using PNML". In *Proceedings of the 2nd WSEAS International Conference on Simulation, Modeling and Optimization (ICOSMO)* (Skiathos, Greece, Sep. 25-28). WSEAS.

Blum, A.; L. Donatiello; P. Heidelberger; S. Lavenberg and E. A. MacNair. 1985. "Experiments with decomposition of extended queuing network models". In *Modeling Techniques and Tools for Performance Analysis*, D. Potier (Ed.). Elsevier Science, 623-640.

Chandy, K. M. and D. Neuse. 1982. "Linearizer: A heuristic algorithm for queuing network models of computing systems". *Communications of the ACM* 25, No. 2, 126-134.

Chandy, K. M.; U. Herzog and L. Woo. 1975. "Approximate analysis of general queuing networks". *IBM Journal of Research and Development* 19, 43-49.

Chatzigeorgiou, A. 2003. "Performance and power evaluation of C++ object-oriented programming in embedded processors". *Information and Software Technology* 45, 195-201.

Courtois, P. J. 1977. *Decomposability: Queuing and computer system applications*, Academic Press.

Heidelberger, P. and K. S. Trivedi. 1983. "Analytical queuing models for programs with internal concurrency". *IEEE Transactions on Computers* 32, No. 1, 73-82.

Jacobson, P. A. and E. D. Lazowska. 1982. "Analyzing queuing networks with simultaneous resource possession". *Communications of the ACM* 25, No. 2, 142-151.

Kahkipuro, P. 2000. *Performance modeling framework for CORBA based distributed systems*. PhD thesis, Department of Computer Science, University of Helsinki, Finland.

Katsaros, P. and C. Lazos. 2002. "Structured performance modeling and analysis for object based distributed software systems". In *Proceedings of the 15th ISCA International Conference on Parallel and Distributed Computing Systems* (Louisville, Kentucky, USA, Sep. 19-21). ISCA, 96-102.

Katsaros, P. and C. Lazos. 2003. "Return state selection for improved effectiveness in sequentially controlled regenerative simulation", (submitted)

Katsaros, P. and C. Lazos. 2003. "A simulation test-bed for the design of dependable e-services", (To appear in the) *3rd WSEAS International Conference on Simulation, Modeling and Optimization* (Rethymno, Greece, Oct. 12-15). WSEAS

Kurasugi, T. and I. Kino. 1999. "Approximation methods for two-layer queuing models". *Performance Evaluation* 36-37, 55-70.

Nikolaidou, M. and D. Anagnostopoulos. 2003. "A distributed system simulation modeling approach". *Elsevier Simulation Modelling – Practice and Theory* 11, No. 3-4, 251-267.

Rolia, J. A. and K. C. Sevcik. 1995. "The Method of Layers", *IEEE Transactions on Software Engineering* 21, No. 8, 689-700.

Sauer, C. H. 1981. "Approximate solution of queuing networks with simultaneous resource possession". *IBM Journal of Research and Development* 25, No. 6, 894-903.

Schweitzer, P. 1979. "Approximate analysis of multiclass closed networks of queues". In *Proceedings of the International Conference on Stochastic Control and Optimization* (Amsterdam, The Netherlands).

Shedler, G. H. 1993. *Regenerative Stochastic Simulation*. Boston, Academic Press.

Vetland, V.; P. Hughes and A. Solvberg. 1993. "Improved parameter capture for simulation based on composite work models of software". In *Proceedings of the 1993 Summer Computer Simulation Conference* (Boston, Massachusetts, USA), 110-115.

Woodside, C. M. 1989. "Throughput calculation for basic Stochastic Rendezvous Networks", *Performance Evaluation* 9.

Woodside, C. M.; J. E. Neilson; D. C. Petriu and S. Majumdar. 1995. "The Stochastic Rendezvous Network Model for performnce of synchronous client-server-like distributed software". *IEEE Transactions of Computers* 44, No. 1, 20-34.

PANAJOTIS KATSAROS holds a BSc degree in Mathematics from the Aristotle University of Thessaloniki - Greece, an MSc degree in Software Engineering from the University of Aston, in Birmingham – UK and a PhD in Computer Science, from the Aristotle University of Thessaloniki. He is a lecturer of Computer Science at the Aristotle University of Thessaloniki – Greece. His research interests are focused on queuing network-based performance analysis of alternate load distribution and fault tolerance strategies, in distributed object systems. He has also published research articles on simulation methodology issues and one book on Programming Language Compilers. e-mail: katsaros@csd.auth.gr
url: http://delab.csd.auth.gr/~katsaros/index.html

CONSTANTINE LAZOS holds a BSc degree in Mathematics from the University of Athens - Greece, an MSc degree in Computer Science from the University of Birmingham - UK and a PhD in Computer Science from the University of Southampton - UK. He was a lecturer for two academic years, at the University of Birmingham and he is a Professor of Computer Science at the Aristotle University of Thessaloniki, since 1980. His published work and research interests are focused on computer systems performance analysis and simulation techniques and he is the author of numerous educational books on Databases, Compilers and Programming Languages (C++, FORTRAN etc). e-mail: clazos@csd.auth.gr

PERFORMANCE ANALYSIS OF AUCTION-BASED ALLOCATION MECHANISMS FOR COOPERATION OF MANUFACTURERS ON PRODUCTION CAPACITY

Barış Tan
Koç University, Graduate School of Business
Rumeli Feneri Yolu, Sarıyer,
34450 Istanbul, Turkey
E-mail: btan@ku.edu.tr

KEYWORDS
Auctions, Cooperation, Markov Chain Models, Performance Analysis, Model Building

ABSTRACT

We present a queuing model to analyze the effects of cooperation of independent producers on production capacity. Producers have limited capacity and have access to subcontractors at a higher cost. When producers agree to cooperate, arriving orders are allocated based on the producers' cost structures and their current loads to meet the due date of the orders. Both single-unit auctions where the arriving order is allocated to one of the producers and also multi-unit auctions where the order is split among the producers are incorporated into Continuous-Time Markov Chain models. These two allocation mechanisms are compared with independent operation of producers without cooperation and also with the centralized operation of producers. Numerical experiments show that a decentralized allocation mechanism that is based on a Vickrey option yields the same average profits for the producers as the centralized operation.

INTRODUCTION

The purpose of this article is to investigate the operation of a productive cooperation network by using a queuing model. An association of autonomous business units operating jointly has been referred as *Agile Enterprise, Joint Venture, Value-Adding Partnership, Virtual Corporation*, etc. in the literature. We use *network organization* to refer temporary cooperation of a number of independent companies to benefit from an opportunity in a market that neither of the firms can exploit on its own (Van Alstyne, 1997).

We consider a market with a number of independent small- to medium-size producers and a single large buyer. The production capacity of each manufacturer is limited and they use subcontractors at a higher cost to produce large orders. In this paper, we investigate the following question: *can independent manufacturers benefit if they cooperate on production capacity?* A following question is if they decide to cooperate, *how can an arriving order and its associated profits be allocated among the producers?*

The motivation of this study comes from textile-apparel-retail channel. A large buyer such as Wal-Mart, Federated, GAP, etc. deals with many small- to medium size enterprises (SMEs) of textile and apparel goods all over the world. In order to be more competitive in global markets, a number of small- to medium-size companies form a network Organization and pool their support activities such as IT, human resources, accounting, and from joint purchasing of raw materials. The network organization also acts as a marketing and sales division of its members to attract large orders from major retailers. For all the other smaller customers, companies act as independent parties and they do not have any obligation to share their customers and orders with the network. When a large order arrives, it is communicated to all the members and allocated to one or more of the members.

We present a queueing model to compare auction-based allocation mechanisms with independent and centralized operation of producers in a dynamic setting. We first examine the mechanisms how an arriving order is to be distributed among the members depending on the volume, capacities, current loads, and the cost structures of the member producers. We also examine how the profit generated from the order should be distributed among the members.

The main contribution of this study is to incorporate auction-based allocation mechanisms in a stochastic model to evaluate the performance of producers with and without cooperation on production capacity. The queueing model allows us to evaluate the performance of auction-based allocation mechanisms in a supply chain with capacity-constrained suppliers that have two modes of production with subcontractors and with randomness in production and arrival of orders.

PAST WORK

In recent years, a number of studies that investigate auctions in operations management applications appeared in literature. We limit our discussion to papers that discuss auctions in a supply chain setting especially for procurement and performance evaluation of auction-based systems. For a thorough review of auction theory, the reader is referred to the book of Krishna (2002) and the review of Klemperer (1999).

Most of the studies that focus on reverse, or procurement, auctions consider a case with one buyer and multiple competing suppliers. In contrast to our study, the problem investigated in the above mentioned studies is to design an auction that maximizes the buyer's utility by deciding on the supplier that will be chosen. A list of related papers is given in the references section.

Another set of papers investigate the performance of systems that use auctions in mostly dynamic settings by using queueing models. For example, Vakrat and Seidmann (2000) investigate the effects of the bidders' arrival process

on the online auctions by using a simple model. Veeramani and Wang (1997) provide a model to evaluate the performance of auction-based distributed shop-floor control schemes with a method that combines approximate queueing network analysis and simulation. In a similar setting, Nandula and Dutta (2000) use Petri nets to evaluate the performance of a manufacturing system that uses auctions as a control strategy.

ALLOCATION MECHANISMS

We are considering a case where orders of Q units arrive randomly to a set of N independent producers. The fixed price of each unit is χ. The producers have limited capacity and they use subcontractors if they cannot meet the due date of the orders by using their regular capacity.

We investigate the possibility of cooperation among the producers to meet the demand together. According to this cooperation agreement, they agree on receiving the orders jointly, possibly by using a central marketing department or a Web site that collects the orders. Then these orders and the profits from the order are allocated among the producers.

This setting is common in procurement auctions. In business-to-business electronic commerce for procurement, it is common to have many suppliers and fewer buyers. In most of the procurement auctions, a single company is engaged in transactions with a number of suppliers.

Auction mechanisms are used to allocate an arriving order. When an order arrives, the producers bid their minimum prices to produce the order. We consider two cases. In the first case, order is not split. Therefore, producer i bids its lowest price b_i to produce the whole order. Then the whole order is allocated to the producer who bids the lowest price.

In the second case, the order is split among the producers. In this case, producer i is asked to submit a bid vector to indicate how much the producer is asking to produce each additional unit. Then the order is split among the producers to have all the units produced at the lowest cost.

We compare these two cases with the independent operation of producers and also with the centralized case where the producers operate as a single entity. The latter case is used as a reference since it is not possible to achieve a better coordination and higher profits. Figure 1 depicts these four different cases for two producers.

When an order arrives, it must be allocated among the members of cooperation. We denote the number of units producer i is allocated by A_i. If the whole order is to be allocated to producer j, then $A_j = Q$ and $A_i = 0$ for $i \neq j$. However, if the order is to be split among the producers, $A_1 + A_2 + \ldots + A_N = Q$.

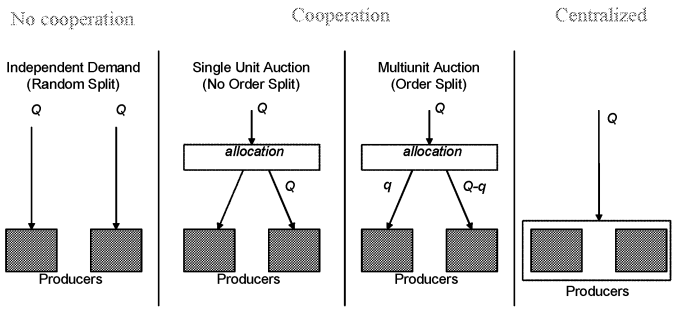


Figure 1. Comparison of Different Cases

The centralized method suffers when members bid lower costs than their actual costs in order to get a higher share from the generated profit. If a member bids with zero-price for the production, then it collects all the profit. If all the members follow the same strategy of bidding zero-cost, then all the bids will be the same and the order is split equally among the producers. The main motivation to introduce auction-based allocation mechanisms is to find a remedy for the above mentioned situation.

Decentralized Allocation and Auction Mechanisms

A buying mechanism has three components: a set of bids, an allocation rule, and a payment rule. The set of bids include all the bids from the buyers. An allocation rule gives the probability that a buyer gets a given number of items depending on the set of bids. A payment rule determines for each buyer, the expected payment as a function of all the bids.

Single-Unit Auctions

In single-unit auctions, the most commonly used auction types are first-price and second-price sealed bid auctions. In a first-price sealed-bid auction, the lowest bidder gets the order and pays the amount he bid. If there are ties, each winning bidder has an equal likelihood of being awarded the object.

In a second-price sealed-bid auction, the lowest bidder gets the order and pays the second lowest bid. Similarly, if there are ties, each winning bidder has an equal likelihood of being awarded the object. In a second-price sealed-bid auction, it is shown that the dominant strategy is to bid the actual cost. Therefore, using a second-price sealed bid auction removes the difficulties associated with allocating more units to a producer who purposely bid a lower cost that its actual costs. Moreover, the expected total cost in both of these auctions is the same (Krishna, 2002).

Multiple-Unit Auctions

We now consider the case where the order is split among a subset of bidding producers. A total of NxQ bids are collected and the Q units are awarded to the Q lowest of these bids. Namely, if bidder i has $A_i \leq Q$ of the Q lowest bids, then i is awarded A_i units.

The pricing of the allocated units can be set in different ways. In a uniform-price auction all Q units are sold at a

market-clearing price that is the highest losing bid. In a Vickrey auction, a bidder who wins A_i units pays the A_i lowest losing bids of the other bids not including his own. In a Vickrey auction, it is shown that it is a weakly dominant strategy to bid truthfully according to real production costs. Furthermore, the Vickrey auction allocates the objects efficiently (Vickrey, 1961).

Once the payment to each producer is determined based on the set of bids, each producer who is allocated a number of units receives a payment that is greater than or equal to what she bid, or equivalently greater than of equal to her production costs. Moreover, an extra profit is also generated for the cooperation. We will use a scheme where this extra profit is to be distributed among the bidders periodically based on the fraction of the total allocated units. It is assumed that this additional future payment is not taken into account by the producers when they bid. In fact, in the examples mentioned in introduction, this extra profit is used to cover the operational expenses of the network organization.

A QUEUING MODEL FOR TWO PRODUCERS

The above discussion was presented in a static setting. We now consider a dynamic setting where orders arrive randomly and distributed among the producers. In this case, the current load of the producers affects their cost structure and therefore their bids for the next order. Then allocation of this new order changes the loads of the producers and therefore affects their cost structure for the next order. In this setting, the performance of the allocation mechanisms needs to be evaluated in a dynamic setting.

Model Assumptions

We consider a system with two producers. The production time of a single product at producer i is an exponentially distributed random variable with mean μ_i. Orders of size Q arrive randomly according to a Poisson distribution with rate λ. There is a fixed due-date of an arriving order.

Production Modes and Cost Structure

Producers have two modes of production. They can use in-house production and, if they need, they can use additional capacity through subcontracting or overtime at a higher cost. We assume that this additional source has no capacity restriction. Therefore, a producer can always meet the due-date of an order by using a combination of regular and additional production. The in-house production cost is c_i and the subcontractor cost is o_i per unit, i=1, 2.

We consider a threshold-type subcontracting policy similar to the one presented for a single server with single-unit arrival system by (Bradley, 2002). Accordingly, upon the arrival of allocated units from an order, if the number of parts at producer i exceeds a threshold Z_i, the exceeding parts are sent to a subcontractor that delivers these parts before the due-date while the remaining parts are produced in house.

The number of parts in Producer i at time t is denoted with $N_i(t)$. The total cost of production for an arriving order depends on the producer's current load, the regular and the subcontractor costs, and also on the subcontracting threshold.

The production function of producer i that gives the total production cost of producing x units depending on the number of parts it has n_i when the order arrives is denoted with $f_i(x)$.

State Space

We analyze this system in the steady-state. The state of the system at time t is $S(t)=(N_1(t),N_2(t))$ with $|S(t)|=(Z_1+1)(Z_2+1)$. Due to the assumptions of exponential service time and Poisson arrivals, the process $\{S(t),t\geq 0\}$ is a Continuous-Time Markov process. Therefore, the steady-state probabilities can be determined easily.

The steady-state joint probability function is defined as

$$p(n_1,n_2)=\lim_{t\to\infty} P[S(t)=(n_1,n_2)]$$

Let $\mathbf{p}$ be a 1-by-$(Z_1+1)(Z_2+1)$ row vector of probabilities

$$\mathbf{p}=[p(0,0),p(0,1),\ldots,p(0,Z_2),p(1,0),\ldots,p(Z_1,Z_2)].$$

Let R be the transition rate matrix of the process $\{S(t), t\geq 0\}$. Once R is determined for different allocation mechanisms, the steady-state probabilities can determined from $\mathbf{p}R = \mathbf{0}$ and

$$\sum_{n_1=0}^{Z_1}\sum_{n_2=0}^{Z_2} p(n_1,n_2)=1$$

The state-transitions are governed by the allocation mechanism that is used to allocate an arriving order. Once the allocation of the order is determined, the profit of producers, the extra profit, and other performance measures can be derived from the steady-state probabilities.

Model Builder

The state-space based methods in the literature follow the approach of generating the probability matrix and then analyzing the probability matrix for the performance measures of interest.

In this study we follow an alternative route to generate the state space and the probability matrix automatically. More specifically, our objective is to build the state space automatically for each allocation mechanism when the system parameters are given

A similar approach has been taken to develop the software tool SHARPE (Symbolic Hierarchical Automated Reliability and Performance Evaluator) that allows its users to construct and analyze performance, reliability, availability, and performability models, especially for computer systems (Sahner, Trivedi, and Puliafito, 1996). Stewart (1994) also presents a tool to generate and analyze Markov models of discrete-event systems.

In this method, we start state-space generation from a given state, say, from state (n_1, n_2). Then we determine the allocation of an arriving order depending on the allocation mechanism. For example, if it is a random arrival, then the

state can change either to (min{Z_1, n_1+Q}, n_2) or to state (n_1, min{Z_1, n_1+Q}) with arrival rates $\lambda/2$ to each state. However, in the single-unit auction case, the order is allocated to the one that has the minimum cost of production. For the order-split case, the result of the multi-unit Vickrey auction needs to be evaluated to determine the allocations A_1 and A_2. Then the state changes to (min{Z_1, n_1+A_1}, min{Z_2, n_2+A_2}). Similarly, in the centralized operation the allocation that minimizes the total production cost needs to be evaluated and then the state changes to the new state similar to the previous case.

Finally, we store the index of the current state, the index of the new state, and the transition rate. This procedure is continued with the next state in the state space until all the states in the state space are evaluated. As a result, the state space and the state transition matrix R are built. All the performance measures can be determined from R.

The explicit determination of the state transition matrices for the four cases that are independent operation of producers, cooperation of producers without splitting an order, cooperation of producers with splitting an order, and centralized operation of producers is not given here due to page limitations.

Performance Measures

The main performance measure is the average profit Producer i makes per unit. When an order arrives and allocated to Producer i, an amount is paid to Producer i for the production of the order. Then the extra profit the cooperation makes from an arriving order is the difference between the revenue from the order and the payment to the producers. The average number of units Producer i gets per unit time is denoted by η_i. When all of the extra profit is allocated among the producers, the total profit producer i gets per unit time in the long run is denoted with π_i^*. We also calculate the average cost of producing an order θ to evaluate the efficiency of the industry as a whole.

COMPARISON OF DIFFERENT ALLOCATION MECHANISMS IN A DYNAMIC SETTING

In this section, we compare the random arrival case (Case I), the second-price sealed bid auction with no order split (Case II), multi-unit Vickrey auction with order split (Case III), and the centralized case (Case IV). We focus on producers' profits, profit increase obtained with respect to the uncoordinated random arrival case, the average cost of producing a unit, and allocation of orders. We report only representative cases. The behavior of the system is tested through numerous experiments.

Figures 2- 4 show the effects of various system parameters on the performance of a specific system. The observations from these numerical studies are summarized as follows: A decentralized allocation mechanism that is based on the multi-unit Vickrey auction yields the same average cost and the same total average profit as the centralized operation of producers. When the producers are identical, they increase their average profit compared to the random arrival of orders by cooperating on production capacity. When the producers are identical, splitting the order yields higher profit compared to the case where the order is allocated to one of the producers. When the producers are not identical, allocation of the extra profit must be based on the differences between the producers. The cooperation is effective when the producers' capacities are not heavily utilized.

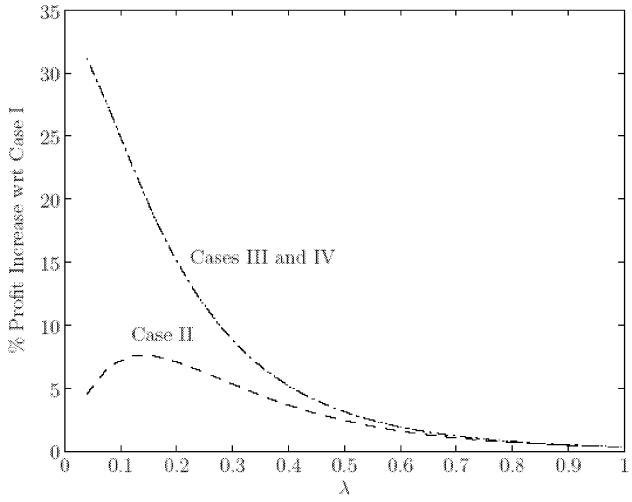


Figure 2. Effect of the Arrival Rate on the Relative Profit Increase of Different Mechanisms with respect to the Random Allocation for Producer 1, μ_1=1, μ_2 = 1, Z_1 = 10, Z_2 = 10, Q = 16, χ= 4, o_1 = 3, o_2= 3, c_1 = 1, c_2 = 1.

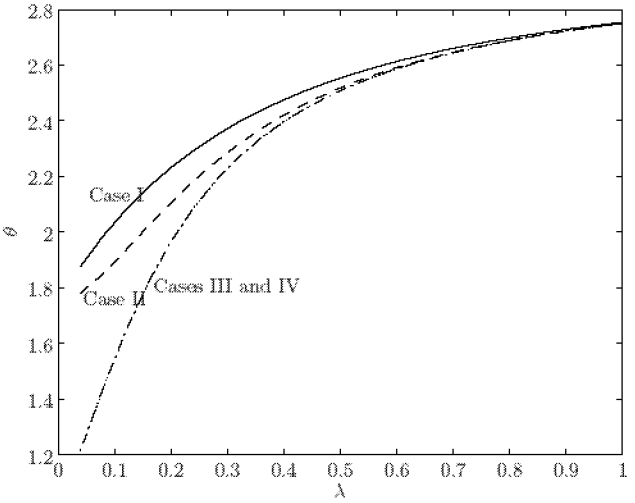


Figure 3. Effect of the Arrival Rate on the Average Production Cost per Unit of an Arriving Order for Different Allocation Mechanisms

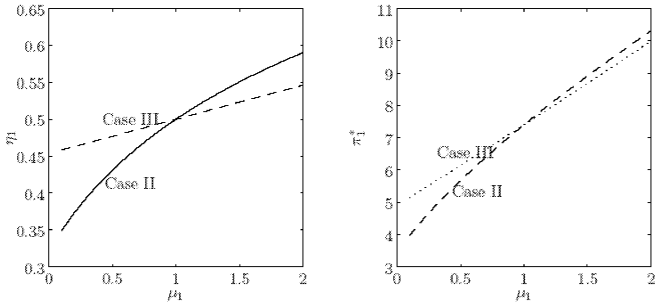


Figure 4. Effect of the Production Capacity on the Allocation of Orders and on Producer 1's Average Profit

CONCLUSIONS

In this paper, we discussed cooperation of producers on production capacity. We propose an auction-based allocation mechanism to allocate orders and profits. A stochastic model is built to analyze these auction-based allocation mechanisms in a dynamic setting. It is shown that cooperation of producers is beneficial for members of the network organization. Especially when the members have similar characteristics and they can not utilize their capacity at high levels, cooperation on production capacity increases their profit substantially compared to independent operation. This benefit comes from using regular capacity of the members more efficiently. In the non-cooperative case, a producer may be forced to use a subcontractor at a higher cost while another producer has available regular capacity.

In order for cooperation to be effective and attractive for all the producers, the allocation of orders, as well as, profits must be managed effectively. We discussed incentive problems for the centralized case. We further showed that decentralized auction-based allocation methods can be used for allocation of orders and profits effectively.

We used a simple queueing model to study an interesting cooperation issue in a production setting with only two producers. The same approach can be used to analyze systems with more number of producers. However, since the method is based on state space generation, it will suffer from rapid increase of states. In this case, the same technique can be embedded in a simulation study and the performance measures can be determined approximately.

REFERENCES

Beil, D. and L. Wein (2002). An inverse-optimization-based auction mechanism to support a multi-attribute rfq process. (http://faculty-gsb.stanford.edu/wein/personal/RFQ-revision1.pdf)

Bradley, J. (2002). Optimal stationary control of a dual service rate *M/M/*1 production-inventory model. (http:// www.johnson.cornell.edu/faculty/profiles/Bradley/)

Eccles, R. and D. Crane (1987). Managing through networks in investment banking. *California Management Review 30* (1), 176–195.

Gallien, J. and L. Wein (2000). A smart market for industrial procurement with capacity constraints. (http://faculty-gsb.stanford.edu/wein/personal/revlastsmartmarketpaper.pdf)

Hamel, G., Y. Doz, and C. Prahalad (1989). Collaborate with your competitors –and win. *Harvard Business Review 67* (1), 133–139.

Huh, W. and R. Roundy (2003). Using auctions for procurement. (http://www.orie.cornell.edu/ huh/singact TR.pdf).

Jin, M. and D. Wu (2003). Procurement auction with supplier coalition: Validity requirements and mechanism design. (http://www.lehigh.edu/ sdw1/jin2.pdf).

Johnston, R. and P. Lawrence (1988). Beyond vertical integration – the rise of the valueadding partnership. *Harvard Business Review 66* (4), 94–101.

Klemperer, P. (1999). Auction theory: A guide to literature. *Journal of Economic Surveys 13* (3), 227–286.

Krishna, V. (2002). *Auction Theory*. California: Academic Press. Kumar, K., H. G. van Dissel, and P. Bielle (1996). The merchant of Prato - revisited: Towards a third rationality of information systems. *Management Information Systems Quarterly*, 1–37.

Mezg´ar, I., G. Kov´acs, and P. Paganelli (2000). Co-operative production planning for small- and medium-sized enterprises. *International Journal of Production Economics 64*, 37–48.

Nandula, M. and S. Dutta (2000). Performance evaluation of an auction-based manufacturing system using coloured petri nets. *International Journal of Production Research 38* (10), 2155–2171.

Sahner, R, Trivedi, K.S., and Puliafito, A. (1996), Performance and Reliability Analysis of Computer systems: an Example-based Approach Using the SHARPE Software Package, Kluwer Academic Publishers, Massachusetts 02061.

Stewart, W.J. (1994), Introduction to the Numerical Solution of Markov Chains, Princeton University Press, New Jersey.

Vakrat, Y. and A. Seidmann (2000). Implications of the bidders' arrival process on the design of online auctions. In *Proceedings of the 33rd Hawaian International Conference on Systems Sciences*, pp. 1–10.

Van Alstyne, M. (1997). The state of network organization: A survey in three frameworks. *Journal of Organizational Computing and Electronic Commerce 7*, 83–151.

Veeramani, D. and K. Wang (1997). Performance analysis of auction-based distributed shop-floor control schemes from the perspective of the communication system. *International Journal of Flexible Manufacturing Systems 9*, 121–143.

Vickrey, W. (1961). Counter speculation, auctions, and competitive sealed tenders. *Journal of Finance 16*, 8–37.

AUTHOR BIOGRAPHY

BARIŞ TAN has a B.S. degree in Electrical Engineering from Boğaziçi University, M.E. degree in Industrial and Systems Engineering and Ph.D. in Operations Research from the University of Florida. He is currently professor of operations management and the director of the Graduate School of Business at Koç University, İstanbul, Turkey. His research interests are stochastic modelling, performance analysis and control of production/inventory systems, and supply chain management.

A new approach to the modelling and analysis of complex discrete systems

A. A. Veselov[1], Chirag Pathak[2], Edmund Kazmierczak[2]
[1] 170006 TSTU, Department of Computer Science, Russia
E-mail: aveselov@tvcom.ru
[2] 111 Barry Street, Carlton, Victoria 3053, Australia
E-mail: {Chirag, Ed}@cs.mu.OZ.AU

KEYWORDS
Distributed model, digital devices, components, interface, protocols.

ABSTRACT

Circuit complexity is increasing as predicted by Moore and the current methods to deal with the state space explosion are already under strain. This paper provides a new way of looking at the state space in order to make it more manageable. The new reachability tree is called a Reachability Tree with a Variable Structure (RTVS). D-PN are used as a formal model for the description of circuits. D-PN were chosen because of the advantages offered by D-PN over conventional Petri Net based models.

INTRODUCTION

Estimation and quality analysis of a systems performance was and still remains a major problem in the design of many different types of systems. The main tool for tackling this problem is state space analysis of all the reachable states in the system. However, constructing the full reachability tree is inextricably linked to the combinatorial state space explosion (SSE) problem. For a complete survey the reader is referred to (Annti and Valmari 1998). The main way of tackling state space explosion is reduction. This however, allows us to deal with the problem in a restricted sense, that is by reducing the size of the model itself and it does not tackle the problem itself. That's why the search for effective ways to deal with the problem is still a major research topic. The paper however, does not make a case for the use of PN as a modeling tool for digital devices. Their use in the modeling and analysis of digital devices has been well established in the literature and goes back to the sixties a detailed description of which can be found in (Patil and Dennis 1972; Hollaar 1982; Varshavsky and Marakhovsky 1996) and Yakovlev provides a complete survey in (Yakovlev A. and A. Koelmans 1998).
The authors purpose two new forms of representing the reachability tree, which will enable us to at least make the SSE more manageable. In the first section we present some modifications to the unified model of the system and some of the special features of these modifications. In the second section we describe the modified view of the reachability tree. It differs from the existing representations mainly in that it forms a structure in which the presence or absence of connections between its nodes is determined by the state of some external signals. Implying that the structure of the reachability tree is not rigid but variable (varies), determined by the set of well-defined external signals. Such a structure is called a reachability tree with a varying structure. In the third section we give another possible way of representing a reachability tree, in this section we also describe the general characteristics and special features of this form. Its main feature is that it includes only static reachable states and excludes the intermediate (dynamic) states. In the last part of the paper we consider a way of combining the new types of reachability trees into a general tree. We conclude by giving a short analysis of the results obtained and give a brief overview of how we plan to use these techniques further.

DISTRIBUTED MODELING OF DISCRETE SYSTEMS AND ITS SPECIAL FEATURES

Practically every distributed system can be represented as a so called "black box" with a set of inputs –X , a set of outputs –Y and a set of internal states –S (fig. 1). However, such a simplified representation is only applicable for very simple discrete systems.

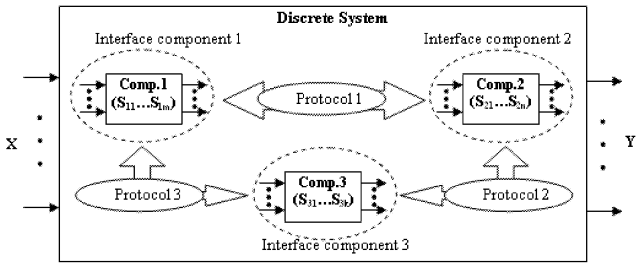


Fig. 1: Simplified representation of discrete systems and its models

Usually complex systems are considered to be composed of some set of more simple, interacting (between themselves) components or subsystems (Fig. 1). Moreover each of these components may be considered as a "black box" with its own input and output signals and internal states and distinguished by its own behavior. Such an approach considerably simplifies the design of a system. So as to interact between themselves within the system boundaries, these components need to have an interface. By an interface we mean a data set available to the components for external use, and carrying out functions. If such data or functions are not available (in other words the component doesn't have an interface) then such a component cannot be used in the system, because in this case it is not possible to arrange any interaction with the

component. A component with zero or null interface represents a fully autonomous subsystem which is not capable of interaction with other components. Hence, it is obvious that in a system that is divided into its component parts (components) there should not be components with an empty interface, because they are not capable of exerting influence on any part of the system and on the system as a whole. The interface is only a tool that enables the components to interact with their environment. In order for the interaction between any set of components to be correct (that is it should not lead to a deadlock situation or collision) it's necessary to develop the appropriate (and in some ways a regulated sequence) interface for signal exchange. We will call such a sequence a protocol. Similar to functional components we can also consider protocols as self-contained/isolated communication components of the system, which in general, can be very complex. In this way, the design process of the system can be seen as consisting of the following stages:

- Dividing the system into parts (components)
- Designing each separate component
- Designing the necessary protocol for organizing the necessary interactions between any set of components in the system

Very often the design of a system starts with the construction of a model, on which the basic technical solutions of the component parts and of the whole system are worked out. The traditional way of modeling the system consists of creating a monolithic model. Certainly this model is also created in parts but the model constructed is such that it directly models the main functioning of each of its parts. On the basis of this we can say that it's more appropriate to construct models of a discrete system so that they are not monolithic, but represent some distributed environment. Such a model is not just physically distributed but each of its components functions as a separate, individual part and receives and influences the remaining component parts of the system through its inputs and outputs. Such a representation of a distributed model allows us to attain parallelism in a more natural way, which is closer to the parallelism in real discrete systems and objects. The attractiveness of using such a distributed model can be summarized below:

- It is no longer necessary to change or carry out the synthesis of the whole system every time we require to carry out some change , even if a very insignificantly small change in some of its component.
- The possibility of implementing a hierarchical model relatively easily.
- A more regular structure for the model itself, which creates the necessary preconditions, for a significant reduction in its implementation and use, for a more wider use and unification and opens up the possibility of constructing a more open system (in the sense that relates to the use of different mathematical tools of modeling) for modeling.

At present Petri nets (PN) are used widely for modeling discrete systems. However the overwhelming majority of the tools use monolithic model specifications. Even developed systems such as CPN-tools (Kurt Jensen 1998). That's why it is advisable to use them only for modeling relatively simple discrete systems. This distributed principle of constructing models will allow ease in modeling systems which are more complex.

Despite significant successes in the development of PN's, researchers continue to search for new, more effective extensions and methods for effective implementation. In this work we have used D-nets (Veselov 2002) as the basis for constructing our models and for illustrating our main points. As this extension is significantly simpler and easier to translate to our domain of discrete systems so it allows us to use and implement the main principles purposed in this work more easily.

D-NET EXTENSION OF PETRI NETS (D-PN).

The original D-net extension to Petri nets was purposed in (Veselov 2002). It represents a specific type of elementary Petri nets with an inhibitor arc and self-loops, and were given the name D-PN(digital extension to Petri nets).The prominent features of D-PN are the ability to represent the modeled objects as an automata, i.e. as a some "black box" with a set of inputs, outputs and internal states. In comparison to classical elementary Petri nets, this extension enables us to model real processes, which occur in a discrete system, more adequately and at the same time interpret the results obtained in the domain area of the modeled objects. The conceptual basis of this extension was made with the following considerations in mind:

- The global state of the system is assessed to be the collection of all its component states. For example, the state of a digital device is evaluated as the corresponding set of voltage levels (states) on all input-output pins of its chips.
- In the modeled device at every particular moment of time only one event can take place. The independence of different events occurring shows up only in the exact time of their occurrence and they are not necessarily mutually dependent, i.e. they can occur independently.
- As a result of an event occurring in the modeled device only one of its signals may change(or is allowed to change) its state.

In general a DPN can be presented in the following way:

A D-PN is a five tuple **DN=(P T F D M)**, where:

P is a finite set of places

T is a finite set of transitions with $\mathbf{P} \cap \mathbf{T} = \emptyset$,

$\mathbf{F=T\times P \rightarrow I=\{}$ **in, out, allow, forbid** $\mathbf{\}}$ finite set of connections (or arcs) between places **P** and transitions **T,** where **I** is the type of the arc which can have one of the four values:

- **In** input arc, represented as an arrow (→), directed from a place to a transition.
- **Out** output arc, represented as an arrow with an inhibitor arc (o→), directed from a transition to a place.
- **Allow** allow-arc, is in the form of a double headed arrow (↔) forming a self loop arc, connecting a place and a transition.
- **Forbid** forbid-arc, is in the form of connecting line starting at a place and ending at a transition with an inhibitor at the end (—o).

$\mathbf{D = T \rightarrow \{ \tau \}}$ time delay on transition firing.

M = P→{ 0, 1 } marking for places, which determines the state of the model.

According to these concepts, which form the basis of D-PN, only four types of basic connections are allowed between its nodes. Input places for transition t_i are represented by **input**(t_i), output places by **output**(t_i), allowed represents its allowing places **allow**(t_i) and forbidding places are represented by **forbid**(t_i). A detailed description of D-PN is provided in (Veselov 2002).
In contrast to ***input*** and ***output*** arcs the forbid and allow arcs don't put a marker into a place or remove it from a place. Their presence only defines the conditions for transition firing. The implementation of output arcs in D-PN represents a structural way of guaranteeing the safety property in this extension.
In addition to this, a restriction is imposed on D-PN which says that no transition can change the marking of more than one place as a result of its firing. That is to say that no transition can have more than one input or output place.

$$\forall t \in T: |input(t)| + |output(t)| \leq 1 \qquad (1)$$

In general all places in a D-PN can be considered as belonging to the following two groups:
- Active places are all places which are associated with the input or output links, these links actually affect the marking of the places.
- Conditional places are all places which are associated with the allow or forbid links, these links don't affect the marking of the places they just either allow an active link to function or forbid it from functioning.

Now equation (1) can be interpreted as saying that each transition may have any number of conditional places but only one active place.
A transition is considered enabled if it has a token in all its allowing places and its input place and it doesn't have a marker in its output place or in its forbidding places.

$$\forall p \in allow(t_i) \cup input(t_i): \quad m(p)=1$$
$$\text{and } \forall p \in forbid(t_i) \cup output(t_i): \quad m(p)=0$$

An enabled transition can fire, if it remains enabled for the time interval, which is determined by ($\tau(t_i)$) for the given transition. As a result of the transition firing the marker is removed from its input place and put into its output place. In comparison to an ordinary Petri Net the execution rules for D-PN differ. When implementing a mechanism for the net's execution, to account for the influence of time on the nets behavior Merlin's (Merlin and Faber 1976) approach is used under the condition that dmax=dmin. The regularity of D-PN structures and the possibility of interpreting its components in digital-electronic terms, enables us to see the circuit implementation of digital devices as a more understandable form for its developers and a special visual form of representing its formal model. In other words, the circuit may play the role of a tool for representing its formal model, analogous to the role played by a graph for a Petri net. For example, people usually find it easier to work with a Petri Net graph rather than a set of matrix equations or some other mathematical formalism of the net which is used later on for the purpose of analysis. This allows us to hide the formal model from the developer, and give him its representation in a form which he is more used to like in this case in the form of a circuit, but the circuits are now live, because this same circuit is also mapped onto a model of the constructed digital device in the form of a D-PN.

Modified representation of a reachability tree

Most of the problems related to the analysis and functioning of modeled objects necessitate the construction of their reachability tree. However the state explosion problem restrains us to studying relatively simple systems with a small no of reachable states only.
The advantages of the distributed model of discrete systems allow us to think about the possibilities of using these main principles for implementation and also for the construction of reachability trees. As already mentioned above, a distributed model is viewed as a system composed of a set of interacting, autonomously functioning components. Hence it's natural that the reachability tree of the whole system should contain the reachability trees of all its components. The usual approach of constructing reachability trees (RT) confirms this. But In this case every node of the sub-tree of the system relating to a component must contain not only the states of that component but also the states of all the other component parts of the system.
However, the general understanding about components in a distributed system is that the models of components are only a part of the system and interact with the system via their inputs and outputs. All the state parts describing the general state of the model in the component-sub-tree are not appropriate for the input and output states of that component and they do not effects the sub-tree in any way. Depending on how big the model is the number of unused states in its component-subtrees increases (or grows). This is how the complexity problem arises.
For reducing the effect of this problem we propose the concept of an enclosed (or compound) reachability tree. The idea of an enclosed (compound) reachability tree is well implemented using D-PN because of its special features, which allow us to select the input signals and its internal states structurally in the model. The main idea consists of the following. For each component of the model we construct a RT, which includes in it not only its internal states but also the states of its outputs and inputs. The output states of the component are seen as states of its environment (as a rule they represent a very small part of the states of the whole system), which render direct influence on the functioning of the component itself. Lets consider the RS-trigger (or flip flop) as an example, which can be considered as a component which is used very often in designing digital circuit devices. In fig.2 we present three different forms of representing such a trigger:

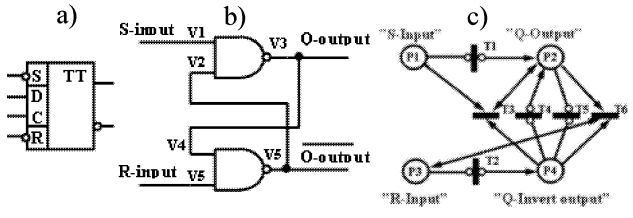


Fig. 2: Circuit representation of a trigger(a), its logical (working) diagram (b) and the D-PN model (c).

- Its graphical image used for circuit representation in the design of systems (fig.2a)
- A logical diagram (fig.2b)
- And its corresponding D-PN model (fig.2c)

The given trigger can be seen as a component which has two inputs 'S' and 'R', through which its environment interacts with it and two outputs 'Q' and 'Q_{inv}', through with the component interacts with its external environment. From a functional point of view a trigger can represents a device with sequential logic or an automaton with memory. Its possible states are formed as a combination of the binary states of four signals: 'S', 'R', 'Q', 'Q_{inv}'. The corresponding complete RT is shown in fig.3a. It is evident that the diagram for the complete RT of the trigger has 16 nodes and 32 connections between them. 16 connections correspond to change of state in input signals (shown in grey lines and are also marked with grey circles) and 16 connections corresponding to the change in internal states (or the triggers own state) states of the trigger (shown in black lines and marked with uncolored circles). A D-PN model of the system does not allow simultaneous change in state of more than one signal because of the firing rule, therefore, the number of connections in the RT of a D-PN model are less than what is theoretically possible. In this respect such a restriction on the D-PN model allows us to construct a more compact RT.

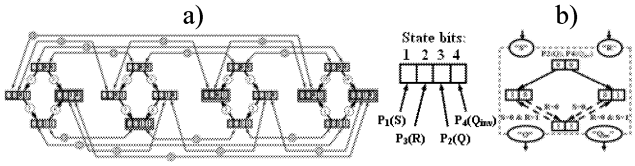


Fig. 3: (a) the standard Reachability tree for a RS-trigger (b) Reachability tree with a variable structure.

As it can be seen from fig.3a, even for such a simple functional device like a trigger, the RT produced is sufficiently complex. However we can easily observe that if we don't consider the direction of the connections between the nodes in the RT then the structure of those parts of the tree which determine the change of only internal states of the trigger (in the figure they are shown in black lines) is the same and does not depend on the states of the external signals 'S' and 'R'. This allows us to continue looking at the RT in the form of a structure which maps change in internal states only and the actual presence or absence of connections between the nodes at every particular (individual) moment is dependent on the state of the input signals. At any particular instance the connections formed can be called the ***operational states*** of the device. In the future we will refer to such a RT as a reachability tree with variable connections or a variable structure (RTVS). The external view of such a tree is shown in fig.3b. Each arc in such a tree describes not only the direction but also some logical functions in which the argument is an input signal of the device. The continuous lines show the constant (unchangeable) connections, for which the logical function of on is always set to a value of true. That is to say that the value is not dependent on the states of the input signals. The dashed lines show the connections between nodes, the presence or absence of which denotes the corresponding logical function. Part of the elements which represent the internal states of the device may be represented simultaneously as its output signals. In this case (fig.3.b) all components of the internal states of the trigger ('P2' and 'P4') are simultaneously also its output signals ('Q' and 'Q_{inv}').

It is evident from fig.3a and fig.3b that the RTVS is significantly more compact in comparison to an analogous RT of a classical form. Such a compact representation is achieved by way of excluding input signals from the number of states which form the general state space of the device and then by representing them in the form of arguments of a logical function, which describe the behavior of the connections between each pair of connected nodes in the RT.

CONSTRUCTING THE COMBINEED REACHABLE TREE WITH A VARIABLE STRUCTURE

By having the RTVS for each functional component of some discrete system we may construct the corresponding RT for it. We will demonstrate this in an example of a more complex device like a RSDC-trigger, the functional (logical) representation and the circuit implementation of the trigger is shown in fig.4a and fig.4b. From fig.4.c we can see that the trigger can be considered to be some device composed of three similar, interacting components, highlighted by dashed lined boundaries in fig.4b. In this case each of these components represents the earlier considered model of an RS-trigger fig.4d.

For constructing the full reachability tree of the RSDC trigger, the initial step is to construct the RT for each of its tree functional components. In this case all three of these components are the same (regular RS-triggers) and they have the same RTVS as already considered earlier in the article.

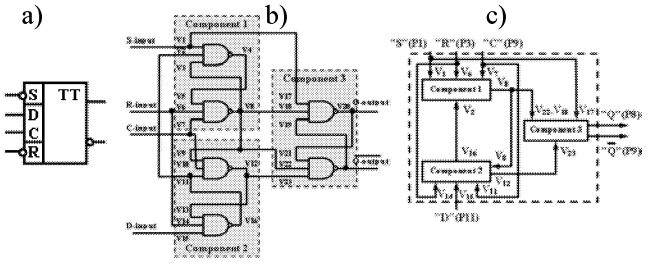


Fig. 4: The logical representation of an RSDC trigger (a), its functional circuit (b) and component schema(c).

That's why we will use the already constructed RTVS for the new component (fig.3b). For describing the general RT for an RSDC-trigger composed of RTVS for each of its functional components which are in the general model we will use the form similar to that used for representing hierarchical models as considered in (Hollaar 1982). The general idea of this representation encapsulated in that the components of the model are seen in the form of some block with a set of inputs and outputs. These pins are used for carrying out the necessary connections between them while forming the general model. In our case the

component's reachability trees (CRT) are similarly considered in the form of a set of such blocks, whose internal composition is represented as a RTVS corresponding to some functional components and not to its model. Inputs of such blocks represent that part of the external environment (surroundings) which is not described by the states of the component itself but only controls the configuration of the RTVS for that particular given component (RTVS). This allows us to obtain a significantly compact form of RT in comparison to its classical counterpart. In figure 5.a we present the distributed D-PN model of an RSDC-trigger on which the dashed lines represent connections between components.

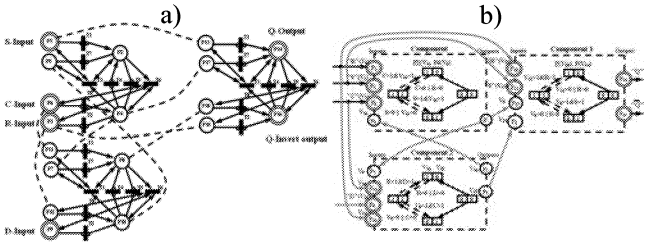


Fig. 5: The distributed D-PN model (a) and its component reachable tree for the RSDC-trigger (b)

If necessary, then from the combinational reachability tree we can obtain the full reachability tree with a variable structure (RTVS). To show this we will consider the combinational reachability tree of an RSDC-trigger as an example (Fig.5). For obtaining the general RTVS for this trigger it is first necessary to obtain the RTVS for any two of its components. Then, we need to combine the obtained RTVS with the second components RTVS. And continue until we don't get the resulting RTVS for the whole trigger. The combination of RTVS components (for example, components 1 and 3) is carried out by replacing each node of the first component (component 1) with the RTVS of the second component (component 3). Then all the connections between the new nodes of the reachability tree of the two components (component 1 and 3) is set up according to those connections, which were in the RTVS of component 1, as shown in Fig.6a. After this, the RTVS obtained in this way is combined with the RTVS of the last component (component 2). The resulting tree with a variable structure is shown in Fig.6b.

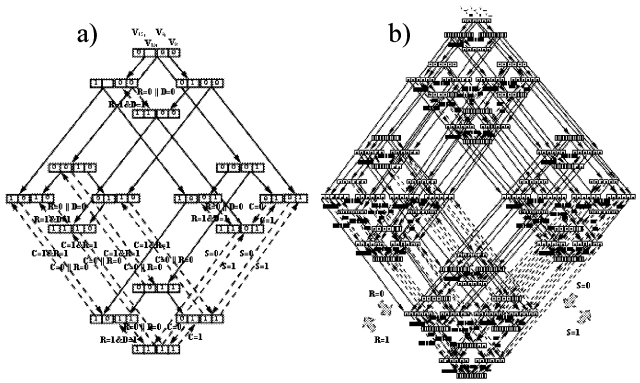


Fig. 6: Intermediate (a) and the full (b) RTVS for the RSDC-trigger.

In Fig.6, similar to Fig.3b, the continuous lines on the RTVS graph show unconditional transitions between its nodes, and the dotted lines represent conditional transitions. We note that, the D-PN model of an RSDCtrigger includes in itself 10 positions (Fig.5a). The complete reachability tree for such a model would comprise of 1024 nodes. The obtained reachability tree for this same model contains 64 nodes (Fig.6.b). The example considered visually shows us the effectiveness of using RTVS.

THE D-PN EDITOR

All the examples in the paper were created in the D-PN editor, which is a tool developed in C++ for the modeling and analysis of Digital Petri Nets. Due to lack of space we will not go into a full length description of the tool but it suffices to say that the D-PN is unique in that it integrates the direct translation of behavioral specification into the circuit implementation, reduction, state space analysis etc. all into one system.

CONCLASION AND FURTHER RESEARCH

A new method for representing the reachability tree the RTVS was purposed and using the RTVS for each component a combined tree was formed. It is hoped that this will make the problem of State Space Explosion more manageable. Much research still needs to be done in exploiting the wide range of possibilities offered by such a representation. This technique can be combined with component level reduction (Veselov 2003) for D-PN and this would make a lot of difference in how we deal with SSE.

REFERENCES:

Annti and Valmari. The State Explosion Problem. In Wolfgang Reisig and Grzegorz Rozenberg, editors, Lectures on Petri Nets I: Basic Models, volume 1491 of Lecture Notes in Computer Science, pages 429-528.Springer-Verlag, 1998.

Hollaar L. A. Direct implementation of asynchronous control units. IEEE Transactions on Computers, C-31(12):1133–1141, December 1982.

Kurt Jensen. Colored Petri nets. Basic concepts, analisis methods and practical use/ Vol.1, Second ed. Springer-Verlag Berlin Heidelberg New York. 1997.

Merlin P. and D.J. Faber. Recoverability of communication protocols. *IEEE Transactions on Communications*, COM 24(9), 1976.

Patil S. S. and J. B. Dennis. The description and realization of digital systems. In Proceedings of the IEEE COMPCON, pages 223-226, 1972.

Varshavsky V. I. and V. B. Marakhovsky. Asynchronous control device design by net model behavior simulation. In J.Billington and W. Reisig, editors, *Application and Theory of Petri Nets* 1996, volume 1091 of Lecture Notes in Computer Science, pages 497–515. Springer-Verlag, June 1996.

Veselov A.A. D-extended Petri nets for simulating of digital devices. Lecture Notes in Informatics (LNI) – *Proceedings series of the Gesellschaft fur Informatik* (GI). Volume P-21 / Jorg Desel(ed.), Mathias Weske(Hrsg). Prozessorienierte Methoden und Werkzeuge fur die Entwicklung von Informationssystemen – Promise 2002. 9-11 October 2002 in Potsdam. GI, 2002. pp. 116-128.

Veselov Alexei, Pathak Chirag. Structural and Behavioral Properties of D-PN extension of Petri Nets. –2003. (to appear soon).

Yakovlev A. and A. Koelmans. Petri nets and Digital Hardware Design Lectures on Petri Nets II: *Applications. Advances in Petri Nets*, Lecture Notes in Computer Science, vol. 1492, Springer-Verlag, 1998, pp. 154-236.

ANALYTICAL SIMULATION OF ELECTRONIC CIRCUITS

PERFORMANCE ASSESSMENT OF TWO D/A MODELS WHEN OPERATING ON TELECOMMUNICATION SIGNALS

Leopoldo Angrisani
Dipartimento di Informatica e Sistemistica
Università degli Studi di Napoli "Federico II"
Via Claudio 21, 80125 Napoli,
Italy
E-mail: angrisan@unina.it

Massimo D'Apuzzo
Dipartimento di Ingegneria Elettrica
Università degli Studi di Napoli "Federico II"
Via Claudio 21, 80125 Napoli,
Italy
E-mail: dapuzzo@unina.it

Mauro D'Arco
Dipartimento di Ingegneria Elettrica
Università degli Studi di Napoli "Federico II"
Via Claudio 21, 80125 Napoli,
Italy
E-mail: darco@unina.it

KEYWORDS
Digital-to-analogue converter (D/A), Integral non-linearity (INL), spread spectrum signals, channel power, occupied bandwidth.

ABSTRACT
The paper pays attention to the role played by the high-speed digital-to-analogue converter in the generation chain of telecommunication signals. In particular, spread spectrum signals are considered and changes and distortions due to non-ideal characteristic of D/A converters are highlighted. The performance of two different D/A models accounting for actual converters functioning is assessed.

INTRODUCTION

Several proposals of 3G (third generation) telecommunications systems, which are mandated to gain a worldwide wireless access, adopt spread spectrum signals. For test and measurement purposes it would be extremely valuable to have at one's disposal spread spectrum signals characterized by known values of specific parameters, namely of those involving the integration of signal power spectrum on a certain frequency range, such as channel power, occupied bandwidth, and ACP (Adjacent Channel Power).

The generation of spread spectrum signals with known parameters is, however, a very difficult task because of their special nature, characterized by a high and variable crest factor (ratio of peak power to average power) and a power spectrum exhibiting white-noise-like features. Anyway, the capability of imposing known values to the parameters characterizing spread spectrum signals is a necessary condition whenever an exhaustive performance assessment of any measurement solution, device and component has to be pursued.

Spread spectrum signals are generated by means of D/A converters. As a consequence, an analysis of changes and distortions in the generated waveform requires the availability of accurate D/A models. This paper aims at evaluating the performance of two of the most recent models proposed in the literature.

PROBLEM STATEMENT

The generation of spread spectrum signals is attained through a multifaceted process based on I/Q modulation techniques. Precisely, two digital baseband signals, named I and Q components, are both multiplied by spreading factors, which produce spread spectrum, and, successively, by scrambling factors, which grant uniform distribution of power in the I/Q plane. The so-modified I and Q components are, then, filtered in order to prevent inter-symbol interference, and impressed upon two carriers running at the same frequency and 90 degrees phase-shifted from each other.

Spread spectrum signals are generated in most cases through hybrid architectures, made up of a digital block followed by an analogue one. The first digitally synthesizes the spread spectrum signal at an intermediate frequency. The analogue block, which is essentially a high-speed digital-to-analogue (D/A) converter, accepts the digital signal in input, and generates it in analogue form. Finally, the (D/A) output is up-converted to the appropriate carrier center frequency.

High-speed D/A converters play a fundamental role in the generation chain: some problems may arise, in fact, from the conversion in analog form of a numerical signal. The output, analog signal surely undergoes changes and distortions with respect to its original features, thus, the values of the parameters of interest, such as channel power, occupied bandwidth, and ACP, turn to be modified; some reference attributes could be lost.

Distortions can be caused by both the quantization process, which takes place at the input of the D/A converter due to its finite resolution, generally expressed in terms of number of bits, and the electronic functioning of the converter itself. Concerning quantization process, it has been verified, through a number of numerical tests carried out on spread spectrum signals with known values of the parameters of interest, that, if the number of bits is greater than 8, the relative difference between the results, in terms of channel power, occupied bandwidth and ACP, obtained from the quantized version of the signal and the original one, characterized by a quasi-infinite precision

(depending on the type of processor adopted), is inside few tens per million.
The main problems related to signal integrity are rather connected to the electronic functioning of the converter itself. Hence, the attention should be mainly paid to the modifications produced by the electronic circuitry of actual high-speed D/A converters.

D/A MODELS

An ideal D/A converter presents a perfectly linear relationship between the digital input (code) and the output voltage, while an actual converter may be affected by linearity and intermodulation errors.

Linearity errors are due to non-perfect weights associated to individual bits.

Intermodulation errors occur, in stead, when the contribution of erroneous bits to the resulting output voltage varies according to the state of the other bits.

A parameter largely used to highlight performance of D/A converters is Integral Non Linearity (INL) that is the difference between the ideal linear characteristic and that really exhibited by the converter.

The knowledge of INL permits to describe, by means of suitable analytical models, the actual D/A converter operation, and, thus, the distortions due to its electronic functioning. In this paper, two different analytical models, namely (Bruce-Stubberud 2000) and (Vargha-Schoukens-Rolain 2001), both presented in the literature, have been taken into account.

The first model, writes the output D/A voltage $y(n\tau)$ adding up the results of a linear transformation $T_l(.)$ and a non linear transformation $T_{nl}(.)$ of the digital input $x(n)$:

$$y(n\tau) = T\left[x(n)\right] = T_l\left[x(n)\right] + T_{nl}\left[x(n)\right]. \qquad (1)$$

It permits, by means of a Fourier technique, to distinguish the spectrum of the output signal in two terms: the spectrum due to a perfectly linear converter and the distortion term due to non-zero INL.

The second one, describes the output analogue signal summing to the response of a perfectly linear converter the contributions of linearity errors (ε_i) and intermodulation errors (ε_{ij}, ε_{ijk},…, $\varepsilon_{1\ldots N}$), suitably estimated on the basis of the measured INL. In particular, the output voltage V_{out} of an N bits D/A converter is written:

$$V_{out} = V_{ref}\left(\sum_{i=1}^{N} \frac{b_i}{2^i} + \sum_{i=1}^{N} b_i \varepsilon_i + \sum_{i=1}^{N}\sum_{j=1}^{N} b_i b_j \varepsilon_{ij} + \right.$$
$$\left. + \sum_{i=1}^{N}\sum_{j=i}^{N}\sum_{k=j}^{N} b_i b_j b_k \varepsilon_{ijk} + \ldots + b_1 \ldots b_N \varepsilon_{1\ldots N} \right), \qquad (2)$$

in which b_i are the binary digits representing the input code.

Due to the long test time, required for the estimation of linearity and intermodulation errors, modeling the device error response using a reduced order model, which needs a limited number of parameters, can be advantageous for testing purposes. This give good reason for introducing the simplified model

$$V_{out} = V_{ref} \sum_{i=1}^{N} b_i \frac{\left(1+\varepsilon_i\right)}{2^i}, \qquad (3)$$

in which only the linearity errors are taken into account.

PERFORMANCE ASSESSMENT

The first D/A model, (1), has been only theoretically proven, while the latter, (3), has been also experimentally validated, but only for slowly varying signals. Thus, the applicability of both models cannot be straightforwardly extended to high-speed D/A converters triggered by fast running timing clocks.

The results of the experimental analysis, presented in this section, account for the capability of the model (1) and the reduced order model (3) of foreseeing changes and variations in analogue spread spectrum signals with respect to the digital version given in input to the D/A converter. The experimental analysis has been executed by means of a suitable measurement station, made up of a workstation, an arbitrary waveform generator (D/A converter, 12-bits nominal vertical resolution), and a digital storage scope (A/D converter, 8-bits nominal vertical resolution, 8GSample/s maximum sample rate, 32 Mbyte memory size).

In particular, to develop a significant analysis, suitable test arrangements have been prepared, taking into account the functional specifics of both the D/A converter and the A/D converter, adopted for measuring the INL of the first.

INL estimation

INL estimation needs the measurement of errors affecting each admissible D/A output voltage. To this aim, an analogue to digital (A/D) converter provided with a number of bits greater than that characterizing the D/A converter under test is required. Furthermore, it is worth highlighting that these measurements should be performed when the D/A converter is triggered at the same generation frequency used to generate the spread spectrum signals, which reaches up to some tenths of megahertz.

At present, the modern technology makes available fast D/A converters that can offer superior voltage resolution with respect to an A/D converter operating at the same clock-rate. As a consequence, there are poor possibilities of measuring the INL exhibited by top-class D/A converters in dynamical conditions. Anyway, high performance A/D converters can grant sufficient measurement speed and vertical resolution for analyzing D/A conversion processes characterized by low vertical resolution, such as those performed by D/A converters embedded in several commercial utilities.

Aiming at experimentally analyzing the D/A conversion process, a functional mode thanks to which the number of bits exhibited by the adopted D/A converter can be arbitrary reduced, has been defined. In particular, this functional mode attains m-bits reduction by dividing the whole set of input codes into subsets, made up of 2^m

consecutive codes, and by associating each subset with the output voltage produced by the first code of the subset. Fig.1 roughly sketches the fundamental principle of this functional mode, showing a 6-bits D/A converter that is reduced to a 4-bits converter.

Besides, a suitable acquisition mode capable of granting enhanced vertical resolution to A/D converters operating at very high sample rates has been exploited. This acquisition mode, supported by several digitizers, implies a real-time processing of the acquired samples. In particular, the digitizer is clocked at its maximum clock-rate, whatever the selected sample rate; during each acquisition interval, that is the selected sample rate reciprocal, all samples taken at the maximum clock-rate are averaged; for each acquisition interval, only the average value is retained into the acquisition memory. It can be proven that, regardless of the input signal, the voltage resolution is increased at the expense of a lower bandwidth acquired waveform. Precisely, an oversampling factor, F, that is the ratio between the maximum sample rate and the selected one, can theoretically grant B-bits enhanced resolution

$$B = \frac{1}{2}\lg_2 F \quad . \tag{4}$$

The INL of the D/A converter used in the tests has been measured for different functional modes, ranging from 6-bits to 12-bits vertical resolution, and, for each mode, at different timing-clock rates. Measurements have been performed making the converter produce all admissible output voltages according to random input sequences, and have been repeated ten times changing the random excitation. The average value attained from the ten measurements has been assigned to INL.

Tests carried out at different timing clocks highlight that INL exhibited by the D/A converter varies and causes performance degradation as the timing clock is more and more increased.

INL results have been also further processed in order to evaluate linearity errors to be used in the model for describing the D/A converter output according to model (3).

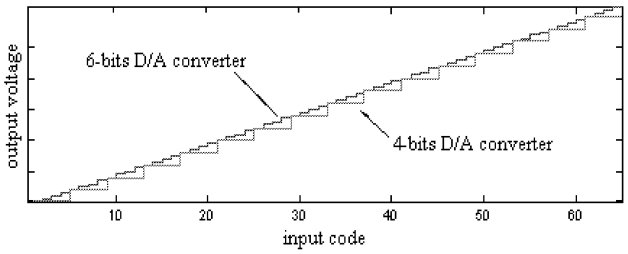


Fig.1 Schematic representation for the functional mode capable of reducing a 6 bits D/A converter into a 4 bits converter.

Theoretical results

The research activity has considered, then, the synthesis of spread spectrum signals. The analysis is described with reference to spread spectrum signals characterized by 10 MHz carrier, 1.667 MHz 99% occupied bandwidth, 0.242 V^2 channel power. The generation frequency is 50 MHz.

The digital spread spectrum signals, synthesized by software tools that use a double-precision numeric representation, thus offering quasi-infinite precision, have been suitably quantized by means of a simulated perfectly linear 10 bits quantizer. Hence channel power, occupied bandwidth, and ACP estimations have been carried out throughout suitable measurement algorithms (Angrisani-D'Apuzzo-D'Arco 2002). The results for channel power and 99% occupied bandwidth, in terms of mean value and sample variance calculated on 50 different realizations, reported in Table I, have been compared to the values calculated on the original signal in order to highlight the effects of the quantization process. No significant effect has been detected. Fig.2 shows the spectrum of the synthesized signal.

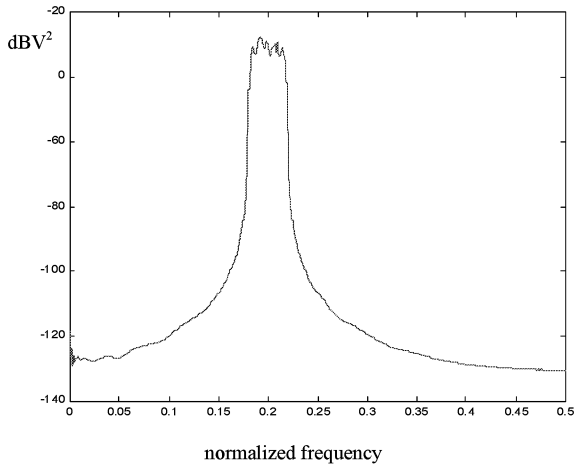


Fig2. Spectrum of the digitally synthesized (quasi-infinite precision) spread spectrum signal, characterized by 50 MHz generation frequency.

Successively, the quantized signals have been passed to the simulated D/A converters, affected by the INL/linearity errors estimated throughout the previous experimental tests. The two theoretical models allow the evaluation of the spectra shown respectively in Fig.3 for the first model and Fig.4 for the second one. It can be noted that, as it was expected, the simplification adopted for the second model, given in (3), causes the failed detection of the spectral growth at 0.4 normalized frequency, which is instead foreseen by the first model, given in (1). Once again, channel power, occupied bandwidth and ACP characterizing the output signal have been estimated. The results for channel power and 99% occupied bandwidth are shown in Table I.

Experimental results

The same quantized signals have been passed also to the internal memory of the actual D/A converter in order to

generate them in analogue form. The output analogue signals have been, then, acquired by means of an A/D converter capable of increasing thanks to filtering and oversampling, the nominal 8 bits resolution with more than 3 bits. The spectrum of the acquired signal is shown in Fig.5.
The experimental results confirm the presence of the spectral growth at 0.4 normalized frequency with respect to the original input.

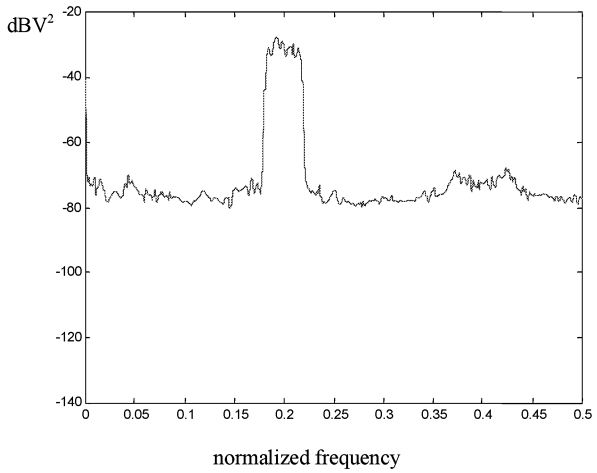


Fig.3 Power spectral density of the spread spectrum signal described by model (1)

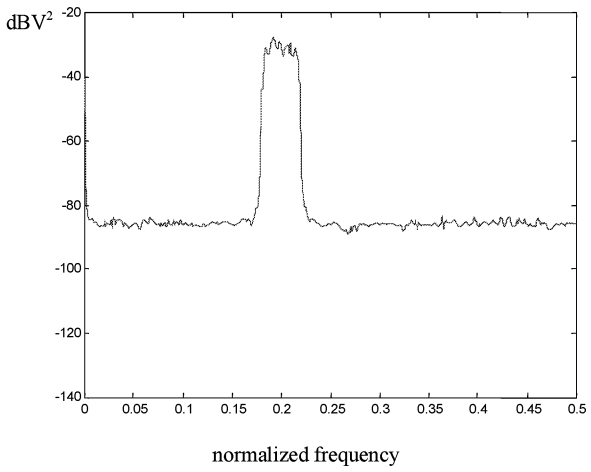


Fig.4 Power spectral density of the spread-spectrum signal described by model (3)

The experimental results shown in Table I highlight that power channel measurements performed on the output produced by model (1) and (3) exhibit a bias with respect to the measured value. Estimation performed by the use of model (1), however, is closer to the measured data, and its experimental standard deviation is inferior to that offered by model (3). On the contrary, occupied bandwidth estimations offered by the two models are compatible.

Furthermore, tests' results also highlight an important side effect, caused by the particular acquisition mode, aimed at improving vertical resolution. The adopted acquisition mode, which achieves resolution enhancement by means of filtering techniques, also rejects uncorrelated noise. As a consequence, the uncorrelated noise produced by the D/A converter under test is under-estimated by the A/D converter. This has been proved throughout several tests carried out by means of D/A converters exhibiting enhanced resolution much greater than that offered by the D/A converter under test. In these cases, the noise floor characterizing the measured spectrum is that of the A/D converter, which is much inferior than that due to uncorrelated noise presented by the D/A converter under test.
This side effect does not occur, in stead, when the enhanced resolution acquisition mode is disabled, as it has been verified by tests carried out on 6-bits and 7-bits D/A converters whose output voltage has been measured by an 8-bits A/D converter functioning in the normal (sample) acquisition mode.
The main fallout of the side-effect accompanied to enhanced-resolution acquisition-mode is that measurements related to the presence of energy in the adjacent bands cannot be afforded; only measurements such as channel power, 95% and 99% occupied bandwidth, are significant.

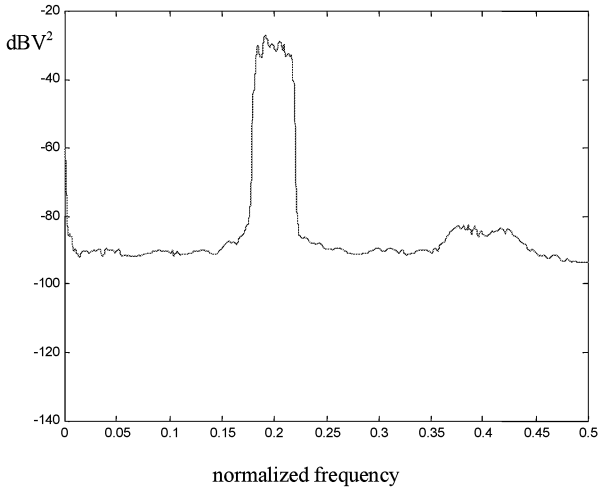


Fig. 5 Power spectral density of the spread-spectrum signal produced by the 10 bits DAC.

Table I

50 realizations	*Channel Power*		*99% occupied bandwidth*	
	average [V²]	*std-dev [%]*	*average [MHz]*	*std-dev [%]*
original input	***0.242***	***2.8***	***1.787***	***4.3***
quantized	***0.241***	***3.6***	***1.789***	***4.1***
model (1)	***0.209***	***3.3***	***1.795***	***6.4***
model (3)	***0.213***	***3.5***	***1.799***	***4.3***
measured	***0.196***	***2.7***	***1.798***	***6.9***

CONCLUSIONS

The capability of accounting for D/A converters functioning, adopted for the generation of spread spectrum

signals, by means of two different analytical models has been assessed. In particular, a comparative analysis of their performance has been carried out throughout an experimental activity.

The experimental activity has involved (i) the characterization of the D/A converter adopted for carrying out the tests, and (ii) the investigation of a suitable A/D conversion process capable of granting sufficient voltage resolution at high sample rates.

REFERENCES

Agilent Technologies 2G & 3G Solutions – Accelerating Progress, Agilent Technologies Literature number 5968—860E, 2002.

L.Angrisani, P.Daponte, M.D'Apuzzo, "A measurement method based on time-frequency representations for testing GSM equipment," *IEEE Trans. on Instr. and Meas.*, vol.49, No.5, Oct. 2000, pp.1050-1055.

L.Angrisani, A.Baccigalupi, M.D'Apuzzo, "Accurate self-synchronising technique for measuring transmitter phase and frequency errors in TDMA digitally encoded cellular systems," *IEEE Trans. on Instr. and Meas.*, vol.53, No.3, June 2002, pp.460-468.

L.Angrisani, M.D'Apuzzo, M.D'Arco, "A new method for power measurements in digital wireless communication systems," in Proc. of *IEEE IMTC/2002*, Anchorage, Alaska, USA, May 21-23, 2002, vol.II, pp.1449-1455.

L.Angrisani, M.D'Apuzzo, M.D'Arco, "Advanced digital signal processing solutions for performance testing and troubleshooting of CDMA transmitters," *Proc. of IMTC/2001*, Budapest, 21-23 May 2001, pp.534-539.

J.W.Bruce, P.Stubberud, "An analysis of harmonic distortion and integral nonlinearity in digital-to-analog converters," Proc. of 42nd Midwest Symposium on Circuits and Systems, 2000, vol.1, pp. 470-473.

C.F.Coombs, C.A.Coombs, *Communications Network Test & Measurement Handbook*, McGraw-Hill, USA, 1998, pp. 417-439.

Design and testing 3GPP W-CDMA base stations, *in Application Note 1355,* Agilent Technologies Literature number 5980-1239E, 2000.

Design and testing cdma2000 base stations, *in Application Note 1357,* Agilent Technologies Literature number 5980-1303E, 2000.

M.B.Priestley, *Spectral analysis and time series*, Academic Press, London, UK, 1981.

D.B.Percival, "Simulating Gaussian random processes with specified spectra," *Computing Science and Statistics*, vol.24, 1992, pp.534-538.

B.D.Ripley, *Stochastic simulation*, John Wiley&Sons, New York, USA, 1987.

Sony-Tektronix, "AWG2000 Series Arbitrary Waveform Generators", User - Programmer Manual.

A.B.Sripad, D.L.Snyder, "A necessary and sufficient condition for quantization errors to be uniform and white," *IEEE Trans. on Acoust. Speech*, Oct. 1977, vol. ASSP-25, No.5, pp.442-448.

B.Vargha, J.Schoukens, Y.Rolain, "Static nonlinearity testing of digital-to-analog converters," *IEEE Trans. on Instrum. and Meas.*, Oct. 2001, vol.50, No.5, pp.1283-1288.

B.Widrow, I.Kollar, Ming-Chang Liu, "Statistical theory of quantization," *IEEE Trans. on Instrum. and Meas.*, April 1996, vol.45, No.2, pp.353-361.

AUTHORS' BIOGRAPHIES

LEOPOLDO ANGRISANI was born in Nocera Superiore, SA, Italy, on April 16, 1969. He received the M.S. degree (cum laude) in electronic engineering from the University of Salerno, and the Ph.D. degree in electrical engineering from the University of Napoli Federico II, in 1993 and 1997, respectively.
Since 2002 he has been Associate Professor at the Department of "Informatica e Sistemistica" of the University of Napoli Federico II. He is involved in research into new methods based on the wavelet and chirplet transforms for detecting, measuring, and classifying transient signals, new methods based on time-frequency transforms for testing RF equipment for mobile communications, new measurement procedures for communication networks test and measurement, and design, realization, and characterization of VXI instruments based on digital signal processors.

MASSIMO D'APUZZO was born in Napoli, Italy, in 1947. He received the M.S. degree in chemical engineering and in electrical engineering from the University of Napoli Federico II, Italy, in 1971 and 1973, respectively.

In 1974, he joined the Institute of Electrical Engineering, University of Napoli as an Assistant Professor of Electrical Measurements. In 1980, he became an Associate Professor of Electrical Measurements, University of Napoli,. Since 1986, he has been a Full Professor of Electrical Measurements at the University of L'Aquila (1986-1988), University of Salerno (1989-1991), and of University of Napoli (1991-present), Italy. His current research interest s are dynamic characterisation of A/D converters and use of wavelet transform for measurement application.

MAURO D'ARCO was born in Salerno, Italy, in 1974. He received the M.S. degree in Electronic Engineering in 1999, and the Ph. D. in Electrical Engineering in 2003.

In 2002, he joined the Institute of Electric Engineering, University of Napoli Federico II, as an Assistant Professor of Electrical Measurements. His research activities are mainly focused on the study and development of measurements methods based on digital signal processing techniques. He also pays attention to the topics concerning signal integrity, and effects of noise and disturbance in electronic systems.

ON THE USE OF MODULATED S-PARAMETERS FOR MODELING RF WIDEBAND AMPLIFIERS

Leopoldo Angrisani
Francesco Falanga
Alessandro Masi
Dipartimento di Informatica e Sistemistica
Università degli Studi di Napoli "Federico II"
Via Claudio 21, 80125 Napoli, Italy
E-mail: angrisan@unina.it

KEYWORDS
Wireless communication systems, RF measurements, RF amplifier characterization, Sinusoidal S-parameters, W-CDMA , Modulated S-parameters.

ABSTRACT

The paper intends to show the inadequateness of classical S-parameters in modelling RF amplifiers when operating with wideband signals such as those involved in 3G communication systems. The use of new parameters, the so called modulated S-parameters, is proposed. Their measurement no longer requires a sinusoid as test stimulus but a digitally modulated signal of the same type the RF amplifier has to operate with. The validity of the new parameters is assessed through a number of time-domain simulations. In particular, an accurate SPICE model of an RF amplifier, addressed to wide-band applications, is first presented. The model gives the opportunity of evaluating both sinusoidal and modulated S-parameters, the ability of which in describing the real performance of the amplifier when operating with W-CDMA signal is compared.

INTRODUCTION

The efficiency demand of wireless communication systems is growing more and more because of their fast spreading on the market and then of competition between the companies involved. The use of instruments allowing effective and fast testing is crucial.
In the 1970s, S-parameters use as a characterization tool of RF amplifiers spread. Although there have been numerous improvements in user control, the fundamental idea of sweeping a little-amplitude sinusoidal signal through a device and measuring its frequency-domain response remains. This methodology worked well up through the 1980s, the age of analogue modulation, but with the advent of complex digital modulation in the early 1990s, designers for digital communication systems began to notice that their traditional test methodologies weren't suitable. There are two fundamental reasons.
A RF amplifier must operate near saturation, to achieve reasonable efficiency. Besides, most modulated signals, like those based on the recent W-CDMA modulation, result in a high *peak to average* ratio (PAR) and power peaks can send the device into strong compression. In these conditions amplifier response to modulated signals varies significantly from the response to sinusoidal signals, this is particularly true for power amplifiers. Figure 1 depicts a segment of a W-CDMA signal [Angrisani L., 2002] applied at the input of an RF amplifier. Figure 2 shows the corresponding output along with the effects of a high PAR; the original peaks of the signal are, in fact, attenuated. As a consequence, classical sinusoidal S-parameters are no longer useful. It is necessary to introduce a new set of parameters, called modulated S-parameters. They are measured providing at input the same signal the amplifier will operate with.

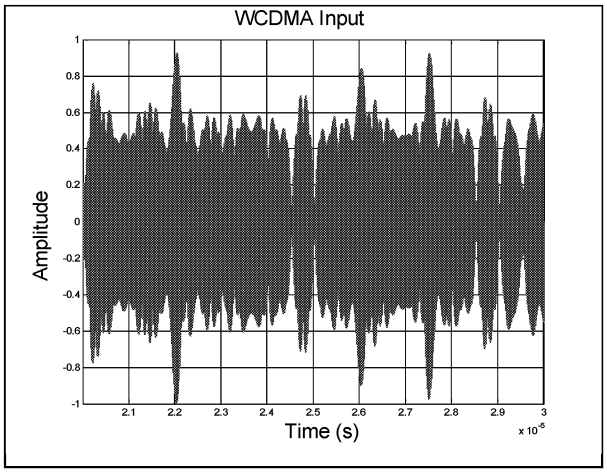


Figure 1: W-CDMA Voltage signal at the input port of amplifier

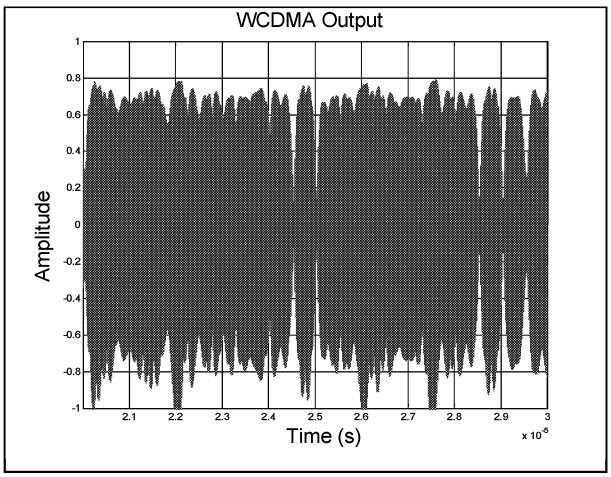


Figure 2: W-CDMA Voltage signal at the output port of amplifier

In the paper, concise definition and explanation of modulated S-parameters are first given. The reliability and

efficacy of these parameters in modelling a RF amplifier used in third-generation (3G) applications are then shown through a number of SPICE simulations [Goyal R., 1988].

MODULATED S-PARAMETERS

S-parameters (scattering parameters) refer to reflected, transmitted and incident waves on a net when it is connected in a transmission line. The easiness with which they are derived makes them a particularly useful instrument to describe transistors and active devices at RF frequencies.
Their classical definition makes use of a, little-amplitude, current or voltage sinusoid. The two port in figure 3 represents the amplifier.

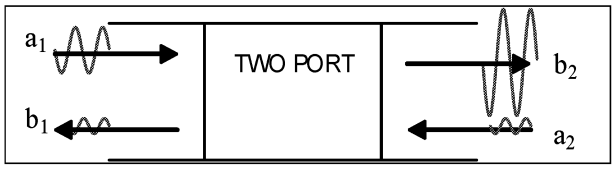


Figure 3: A typical amplifier as a two port

$$S_{11} = \left.\frac{b_1}{a_1}\right|_{a_2 = 0} \quad S_{21} = \left.\frac{b_2}{a_1}\right|_{a_2 = 0}$$

$$S_{22} = \left.\frac{b_2}{a_2}\right|_{a_1 = 0} \quad S_{12} = \left.\frac{b_1}{a_2}\right|_{a_1 = 0} \quad ; \qquad (1)$$

where

$$a_1 = \frac{V_1^+}{\sqrt{Z_1}}\ ,\ b_1 = \frac{V_1^-}{\sqrt{Z_1}},\ a_2 = \frac{V_2^+}{\sqrt{Z_2}},\ b_2 = \frac{V_2^-}{\sqrt{Z_2}}. \qquad (2)$$

Subscript 1 indicates input port, subscript 2, the output port; V^+ is the amplitude of incident signal, V^- is the amplitude of reflected signal, Z_i is the impedance of port *i*. The conditions a_2=0, a_1=0 represent output and input load match. From (1) and (2) the following relations result:

$$|S_{11}|^2 = \frac{P_r}{P_i}; |S_{21}|^2 = \frac{P_t}{P_i} \qquad (3)$$

Where P_r, P_t, P_i are, respectively, the mean power of reflected, transmitted and incident signal. Substantially S_{21} is the gain of the amplifier and S_{11} the input reflection coefficient. Similar considerations can be made for S_{22} and S_{12}. The last representation is very diffused in the data sheets of active devices.
Current instrumentation allows the evaluation of S-parameters from power ratios measurements rapidly and with great accuracy even at RF frequencies.
Modulated S-parameters are measured starting from a stimulus that is no longer a sinusoid but a digitally modulated signal (figure 4).



Figure 4: The two port with W-CDMA signal

Consequently modulated S-parameters are defined as follow:

$$|S_{11m}|^2 = \frac{P_{rm}}{P_{im}}; |S_{21m}|^2 = \frac{P_{tm}}{P_{im}} \qquad (4)$$

Where the mean power is now referred to digital modulated signal (*m* subscript).

THE AMPLIFIER UNDER TEST

The MSA-0986 is a high performance silicon bipolar Monolithic Microwave Integrated Circuit (MMIC) [McCullagh M. J.] housed in a low cost, surface mount plastic package. This MMIC is designed for very wide bandwidth industrial and commercial applications that require flat gain and low VSWR. It is fabricated using HP's 10 GHz f_T, 25 GHz f_{MAX}, silicon bipolar MMIC process [Agilent Technologies].

Features

- Broadband, Minimum Ripple Cascadable 50 Ω Gain Block
- $7.2 \pm 0.5dB$ Typical Gain Flatness from 0.1 to 3.0 GHz
- 3 dB Bandwidth: 0.1 to 5.5 GHz
- 10.5 dBm Typical P_{1dB} at 2.0 GHz
- Surface Mount Plastic Package
- Tape-and-Reel Packaging Options Available

This Monolithic Silicon Amplifiers have an internal structure consisting of a Darlington connected pair of bipolar transistors embedded in a matrix of resistors [Armijo C. T., 1989]. Since this structure is current controlled, the bias point of an MSA can best be described by specifying the total device current i_o (figure 5).

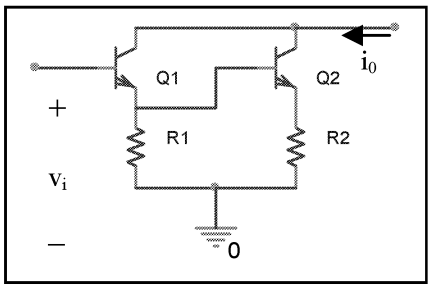


Figure 5: The Darlington configuration

Spice model of MSA-0986

Agilent technology has provided the *netlist* of the Spice model of the amplifier. It lists the SPICE input data file and consists of a number of sub-circuits along with elementary components and their interconnections.
The netlist is hierarchically organized, thus allowing an agile and straightforward representation of each part of the whole circuit of the amplifier. The model also includes the parasitic elements representing the pins and the package of MSA-0986.
For the sake of clarity, each part of circuit is described in the following with reference to its SPICE model (figures 6-11):

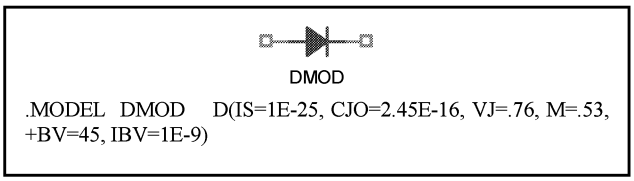


Figure 6: Diode model

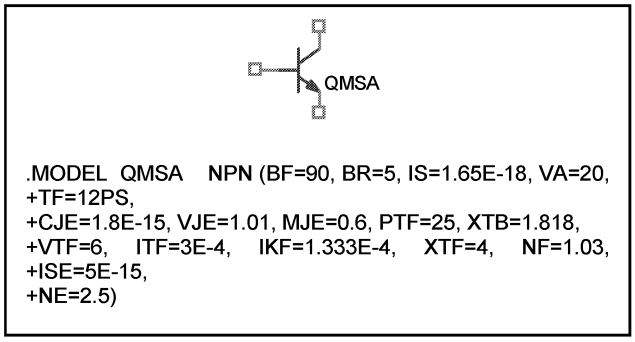


Figure 7: Intrinsic transistor model

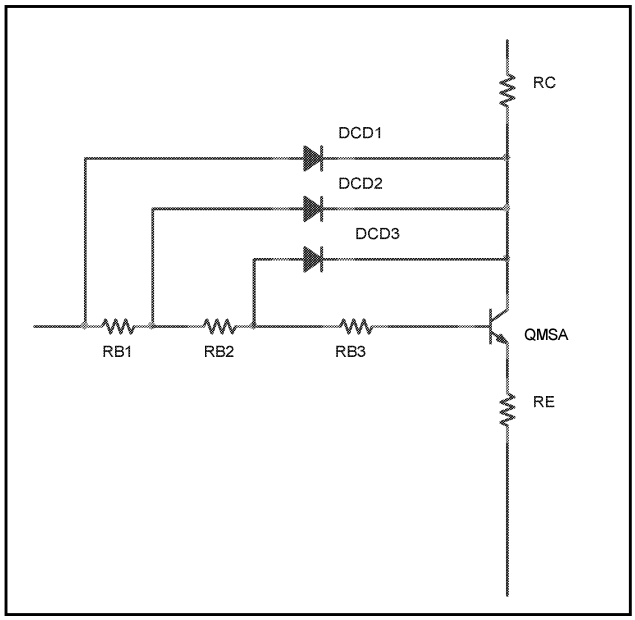


Figure 8: Distributed base transistor model of Q1 and Q2

Figure 9 shows the heart of the amplifier: a Darlington topology with biasing and feedback nets properly dimensioned to provide great gain and bandwidth performance [Stoukatch O. V.].

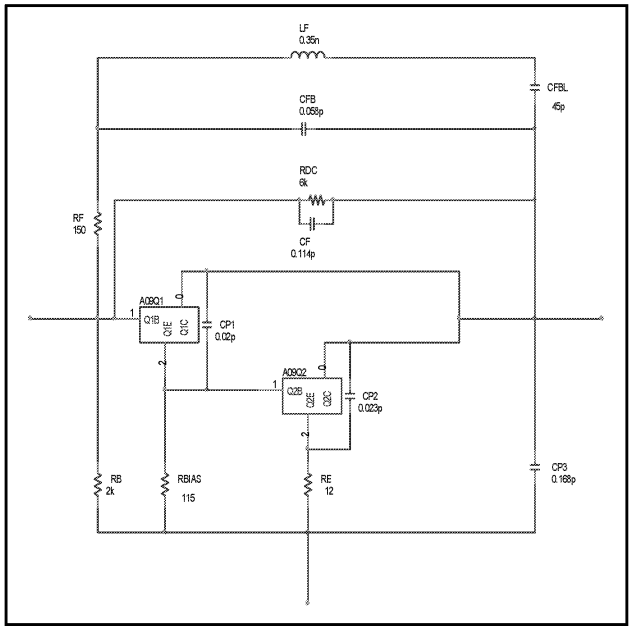


Figure 9: MSA-0986 die model

At operating frequencies of the device we must include, in the model, all parasitic components due to the package: three input, output and ground pins (CxTx,LTx) modelled as transmission lines, parasitic capacitors (CCEB, CCEC, CCBC); contact inductances (LLx), bond wire inductances (LLB, LLE) (figure 10).

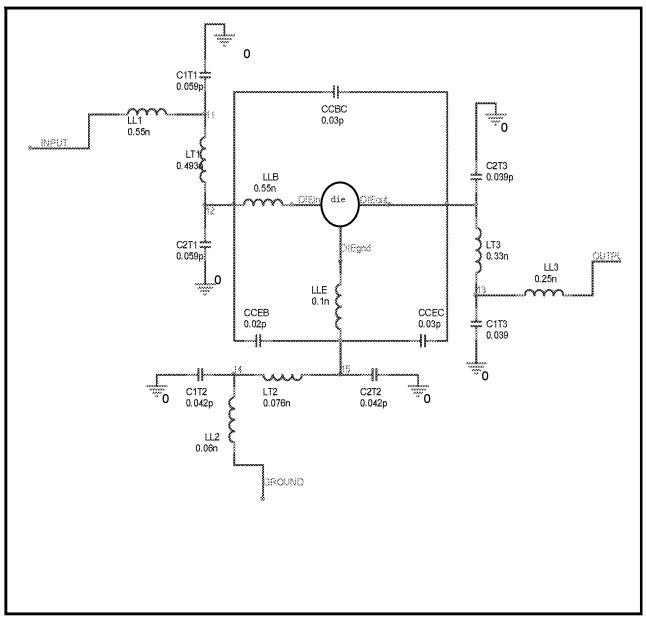


Figure 10: Package model

Recommended operating current of the device is 35 mA; the range of frequencies of our interest is 800 MHz – 2 GHz; to provide a good thermal stabilization a voltage drop of at least 4 Volts is recommended across the series R_{c1}, R_{c2} (figure 11). These considerations brought to the choices, shown figure 11, about the values of discrete components of the test circuit.

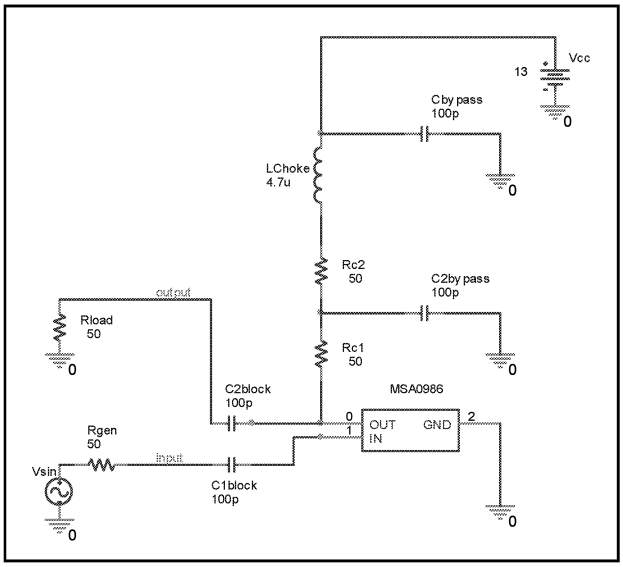


Figure 11: Complete test circuit model

SIMULATIONS

Time domain simulations have been carried out. The scanned frequencies are: 0.8 to 2 GHz with a step of 100 MHz. For each frequency two output files are generated: one to determine S_{11} and S_{21}, where the independent signal generator has been set at the input port of device and the 50 Ω load at the output; one to determine S_{22} and S_{12} where the signal generator has been set on the output, and the 50 Ω load on the input. Table 1 shows the output file layout of each simulation. This contains the time samples of input and output voltage and current, from which it can be derived the mean power at input and output port: P_{input}, P_{output}.

Header1		Header2	
TIME	V(OUTPUT)	TIME	V(INPUT)
5.000E-08	3.880E-02	5.000E-08	-4.625E-04
5.011E-08	-1.333E-02	5.011E-08	2.983E-02
5.022E-08	-5.656E-02	5.022E-08	4.848E-02
…		…	
Header2		Header2	
TIME	I(OUTPUT)	TIME	I(INPUT)
5.000E-08	7.759E-04	5.000E-08	9.250E-06
5.011E-08	-2.666E-04	5.011E-08	5.487E-04
5.022E-08	-1.131E-03	5.022E-08	8.781E-04
…			

Table 1: Output file of spice simulation

The S parameters are then evaluated as follow:

$$S_{11} = \frac{P_{reflected}}{P_{incident}} = \frac{P_{incident} - P_{input}}{P_{incident}};$$

$$S_{21} = \frac{P_{output}}{P_{incident}}. \qquad (5)$$

$P_{incident}$ is banally determined carrying out a similar simulation where the load of the signal generator is a 50 Ω resistance.

The time samples are saved on the output file with a constant time step $T_{step} = \frac{1}{10 \cdot freq}$; this is a good trade-off between results resolution and computational time.

Starting with its definitions, mean power can be easily evaluated as follow:

$$\overline{P} = \frac{1}{T}\int_T V \cdot Idt \Leftrightarrow \overline{P} = \frac{1}{\sum_k \Delta T_k}\sum_k V_k \cdot I_k \cdot \Delta T_k$$

and then

$$\overline{P} = \frac{T_{step}}{n \cdot T_{step}}\sum_k V_k \cdot I_k \Rightarrow \overline{P} = \frac{\sum_k V_k \cdot I_k}{n} \qquad (6)$$

Mean power is nothing but the arithmetic mean of the product of time samples of voltage and current.

Measurement algorithms

With help of simple Matlab routines it has been possible, starting from SPICE output simulation files, to generate final results. Figure 12 shows the flow chart of the procedure which measures the S-parameters: The vector of S-parameters is initialized to zero; then a 'for' cycle is repeated a number of time equals the number of frequencies scanned; Sxx(*m*) gives at *m* frequency the four S-parameter.

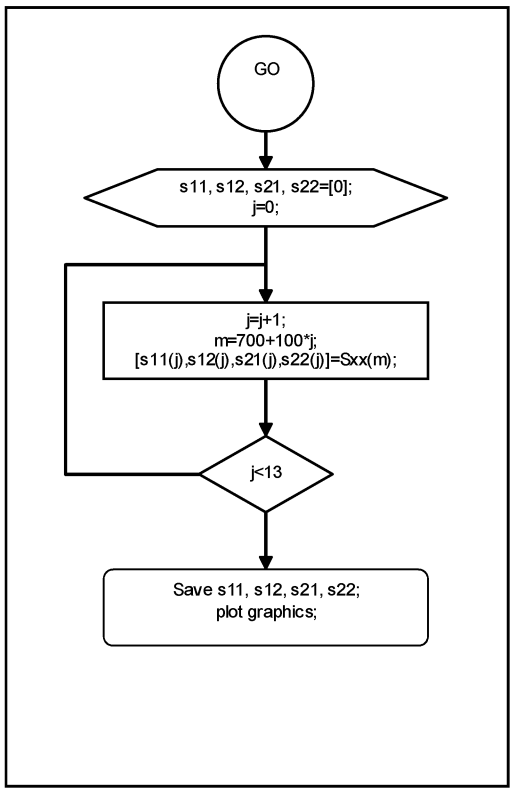


Figure 12: The algorithm for S-parameters measurement

Figure 13 shows the flow chart of Sxx( ) subroutine.

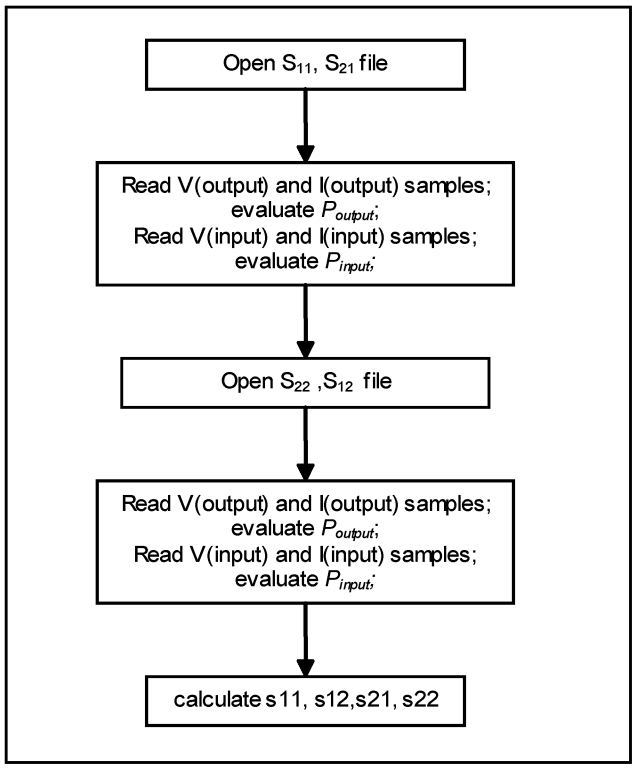


Figure 13: The Sxx().m subroutine

The use of both Spice and Matlab programs allowed us to obtain in a similar fashion the $P_{incident}/P_{output}$ characteristics of our amplifier and find its P_{1dBm}, that is the $P_{incident}$ value where the ideal linear characteristic is far from the real one of 1dBm. Substantially it marks the begin of saturation of the amplifier. This can be made at all frequencies.
In the following the complete measurement steps are presented:

- Determine $P_{incident}/P_{output}$ characteristics.
- Evaluate sinusoidal S-parameters.
- Generate, with Matlab, a segment of a complex modulated signal, the W-CDMA, at all frequencies considered.
- Save these segments in a Spice format to be utilized as independent W-CDMA signal generator instead of the sinusoidal, in SPICE.
- Evaluate modulated S-parameters.
- Compare sinusoidal and modulated S-parameters.

RESULTS

All the results are contained in the following figures from 14 to 24 [Falanga F., 2003].

$P_{incident}/P_{output}$ characteristics of MSA-0986

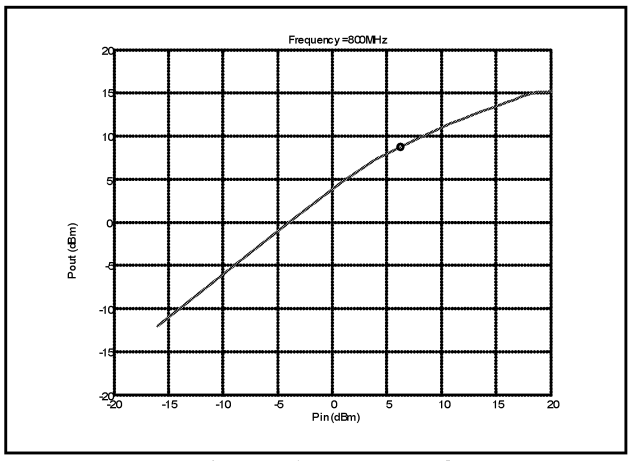


Figure 14: $P_{incident}/P_{output}$ characteristic @ 800MHz

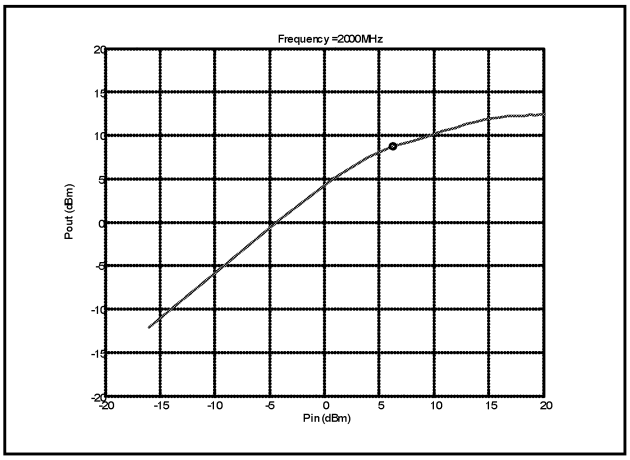


Figure 15: $P_{incident}/P_{output}$ characteristic @ 2GHz

The red circle in figures 14-15 shows the compression point at 1dBm.

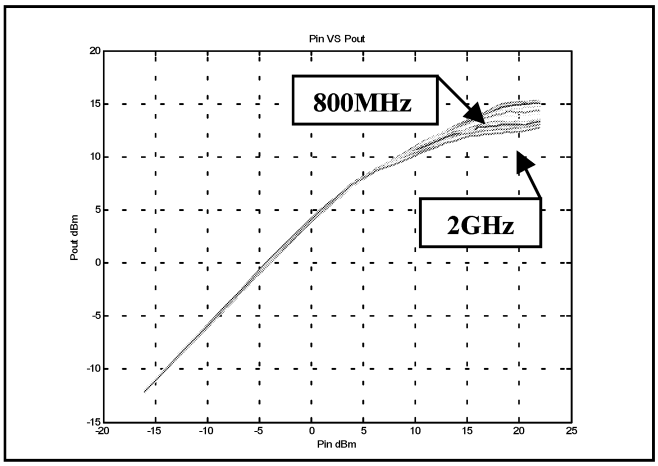


Figure 16: $P_{incident}/P_{output}$ characteristic at all frequencies

Sinusoidal S-parameters and modulated S-parameters both with a mean input power at -16dBm (very far from the compression point)

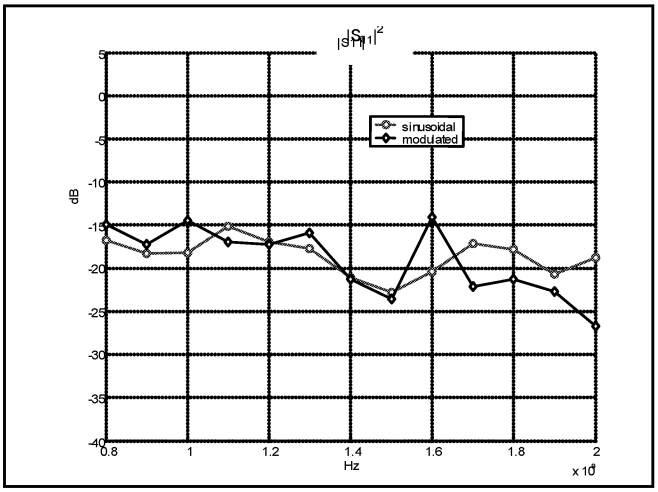


Figure 17: $|S_{11}|^2$

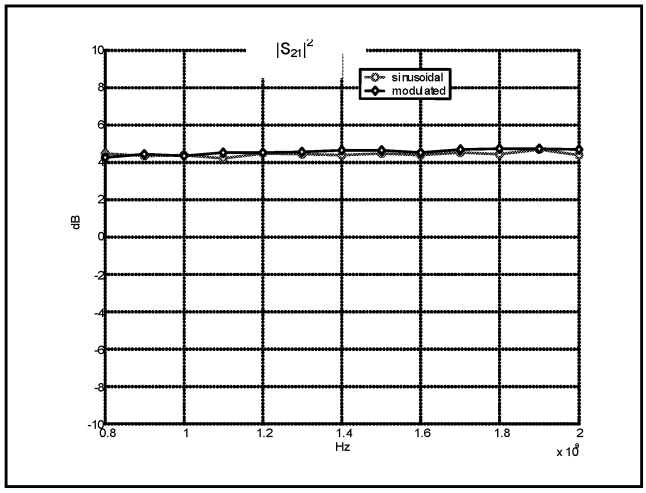


Figure 18: $|S_{21}|^2$

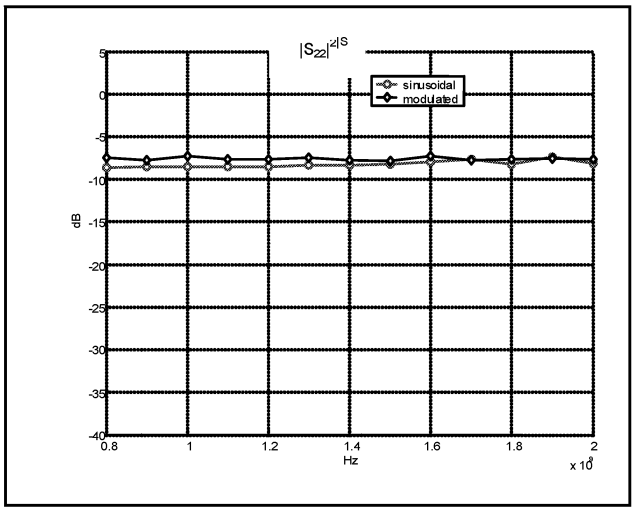


Figure 19: $|S_{22}|^2$

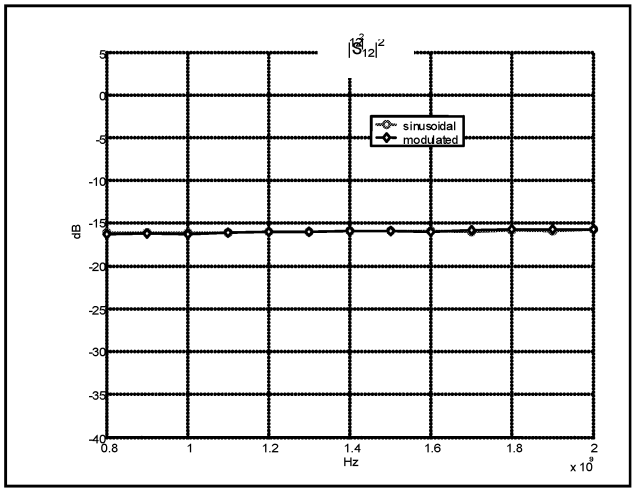


Figure 20: $|S_{12}|^2$

It can be noted how the two sets of parameters are perfectly equivalent when the power of both signal (sinusoidal and W-cdma) is set far from the compression point of amplifier.

Sinusoidal S-parameters and modulated S-parameters with mean power of W-CDMA signal next to the compression point

As the power of W-CDMA incident signal approaches compression point, relevant differences emerge in

- S21: the gain of the amplifier. It is lower in modulated case because of the high PAR (Peak to Average Ratio) of complex modulated signal (figure 22).
- S11: the reflection coefficient. It is higher in modulated case, showing that input match get worst if power level of signal increases [RF Micro Devices]. (figure 21).

These differences could easily mean the difference between a design passing its specifications or failing them.

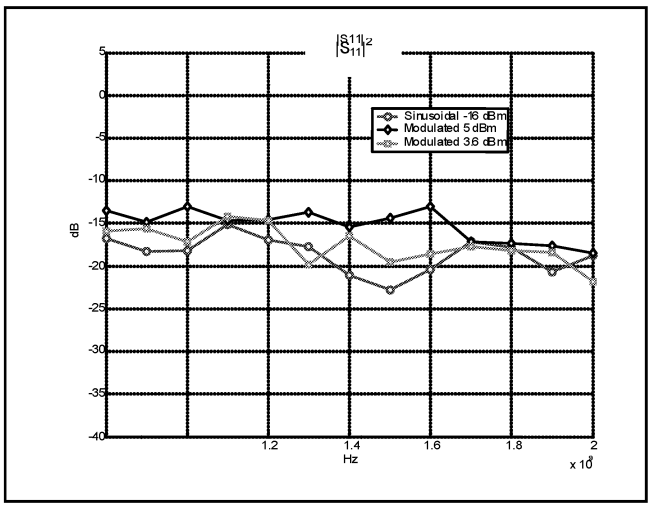


Figure 21: $|S_{11}|^2$

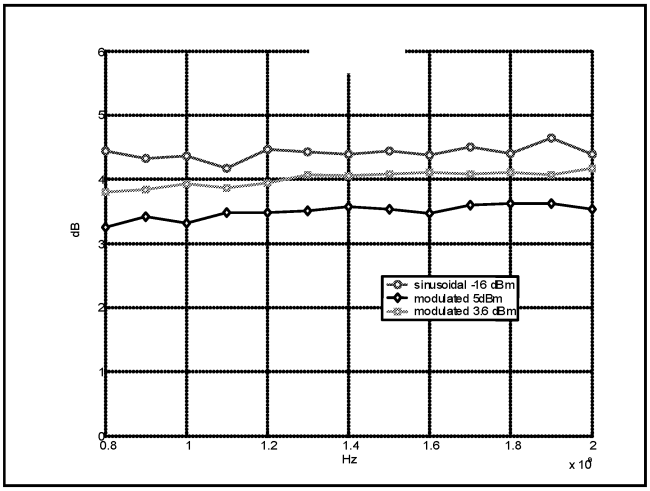


Figure 22: $|S_{21}|^2$

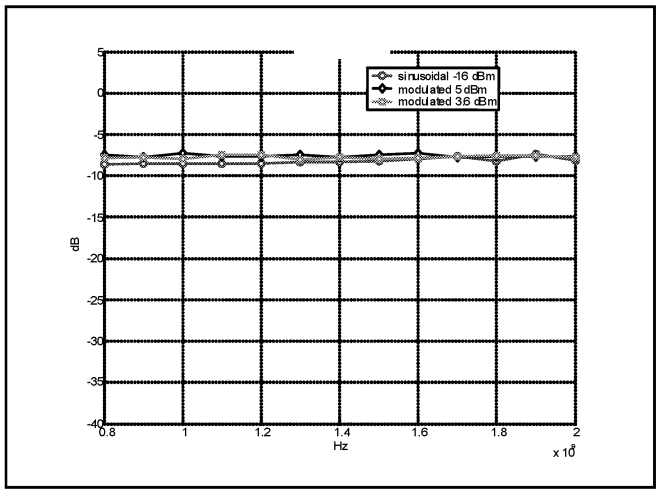


Figure 23: $|S_{22}|^2$

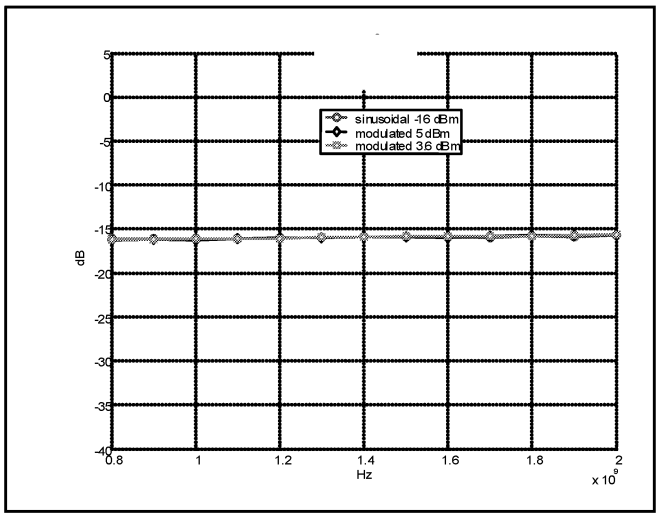


Figure 24: $|S_{12}|^2$

CONCLUSIONS

Developed nearly 35 years ago, S-parameters are essential tools of the RF/microwave designer.
They have worked well, but sinusoidal S-parameter data cannot completely describe active devices subjected to signals having complex wideband modulations.
The coincidence of two S-parameters models in the linear region already gives validity to modulated S-parameters but the relevant aspect is shown in proximity of the compression point of amplifier.
Here sinusoidal S-parameters are no longer adequate to describe the real performance of the amplifier thus modulated S-parameters becomes more representative.

REFERENCES

Agilent Technologies. "Biasing MSA Series RF Integrated Circuits".

Agilent Technologies. "Cascadable Silicon Bipolar MMIC Amplifier".

Agilent Technologies. "INA/MSA RFIC Amplifier Demontration Board", assembly and Operating Instructions.

Angrisani L, D'Apuzzo M., D'Arco M. "A new method for power measurements in digital wireless communication systems", Proc. of IEEE IMTC/2002, Anchorage, Alaska, USA, May 21-23, 2002, vol.II, pp.1449-1455, IEEE Trans. on Instr.and Meas., in press.

Armijo C. T., Mayer R. G. "A new Wide-Band Darlington Amplifier". IEEE journal of solid-state circuits, Vol. 24, No. 4, August 1989.

Goyal R. "S-Parameter Output from the SPICE Program" , Proc. of IEEE CIRCUITS AND DEVICE MAGAZINE, March 1988.

Falanga F. "Validation of RF amplifiers 3G applications dedicated, with the use of modulated S-Parameters" , in Italian, MD Thesis 2003

Matlab is a registered trademark of MathWorks Inc.

McCullagh M. J., Roberts N.B. “HBT power MMIC development at GMMT”.

PSpice is a registered trademark of MicroSim Corp.

RF Micro Devices, Inc. “A study of Device Input Impedance in Pspice”.

Stoukatch O. V. “The Darlington Amplifier with Extended Frequency Band”, IEEE Tomsk Polytechnic University, Tomsk, Russia.

NUMERICAL METHOD USED FOR SIMULATION OF POWER ELECTRONIC CIRCUIT. MODELISATION OF THE SEMI-CONDUCTOR BY PERFECT SWITCHES.

Faouzi BOULOS
Department of Physics
University Lebanese of Science I I - Fanar
BP 90656, Jedeidet-El-Maten, Lebanon
Email: faboulos@ul.edu.lb

KEYWORDS
Numerical modelling techniques, numerical simulation, semi-conductor modelling, power electronic circuits.

ABSTRACT

The aim of this paper is to describe a numerical simulation method and compares the different mathematical methods, techniques and numerical calculations of power electronic circuits where the semi-conductors are simulated by perfect switches. The method consists in simplifying the initial circuit and setting up the mathematical formulation and development of the automatic simulation. A method is shown based on a topological analysis of the circuit allowing the extraction of the relations and the calculation of the voltage across a turned off semi-conductor.

INTRODUCTION

The reliability of a simulation program depends on the choice of the method of numerical calculation and the choice of the electric model of the components of commutation. The problem of the method of numerical calculation is the presence of inductive and capacitive nonlinear elements. One expresses unspecified currents or voltages starting from variables of states: the terminal voltage of the capacitor and the current in an inductance. This can pose problems of convergence with the choice of the step of calculation as in the iterative methods presented in this paper. The choice of the electric model of the semi-conductor is thus very important. There are several ways to represent the semi-conductor which are all electrically equivalent (Antognetti and Massobrio 1988). Similarly, all semi-conductors may be represented by equivalent electric circuits more or less complex, which can be used in the electronic circuit simulation for example SPICE (Nagel and Pederson 1973) Some of the most frequently used models are:
a- binary variable resistance depending whether the semi-conductor is "off " (high resistance) or "on" (low resistance) (Schonek J. 1977), b- controlled voltage source or current source (Laktos 1979), c- inductance in series with a parallel RC circuit (Rajagopalan 1978),d-inductance and resistance (Eisenar and Hofmenister 1972).

In these cases the topology is fixed, the circuit and the size of the calculating matrix is fixed. Another method is to represent the semi-conductor by replacing it by an open circuit when it is "off" and by a short circuit when it is "on". Thus, the topology is variable. Each condition has its set of equations, which reduces the size of the calculating matrix considerably (Boulos 1988). On the other hand in the variable topology method the semi-conductors are simulated by perfect switches, the graph and the equation system are variable (Boulos 1988). The difficulty is due to the fact that in such a method the semi-conductors are simulated by perfect switches, that is two nodes connected by a conducting semi-conductor are merged and two nodes connected by a turned off semi-conductor have their branches removed, sometimes leaving some nodes pending. The branches terminated by pending nodes are also removed. This point is resolved by an algorithm (Boulos 2001); that makes it possible to modify the complete topology of the circuit at each sequence of operation. This makes the knowledge of the current in a conducting semi-conductor and the knowledge of the voltage across a turned off semi-conductor difficult to obtain. In a previous publication (Boulos 2002) I have presented a solution for the method based on a topological analysis of the circuit allowing the extraction of the relations calculating the current in a conducting semi-conductor.
In this paper I present solutions for the extraction of the relations and the calculation of the voltage across a turned off semi-conductor and for the choice of the step of calculation as in the iterative methods.

THE FIXED TOPOLOGY METHOD

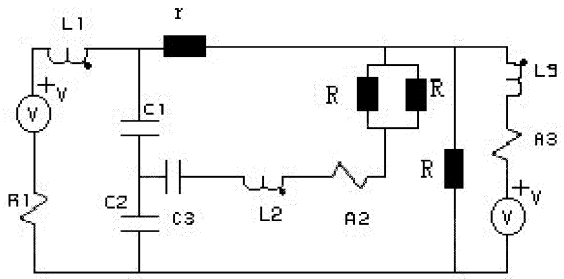


Figure1: The semi-conductor is "off "(high resistance R) or "on" (low resistance r).

In this type of simulation, the circuit is fixed and the equation system is unique, yet, some values of the coefficients may change according to the operating point of the semi-conductors. They are considered high resistance when they are "off" (R) and low resistance when they are "on" (r). For the principal function, see figure 1.

THE VARIABLE TOPOLOGY METHOD

In the variable topology method, the semi-conductors are simulated by perfect switches, shown in figure 2.

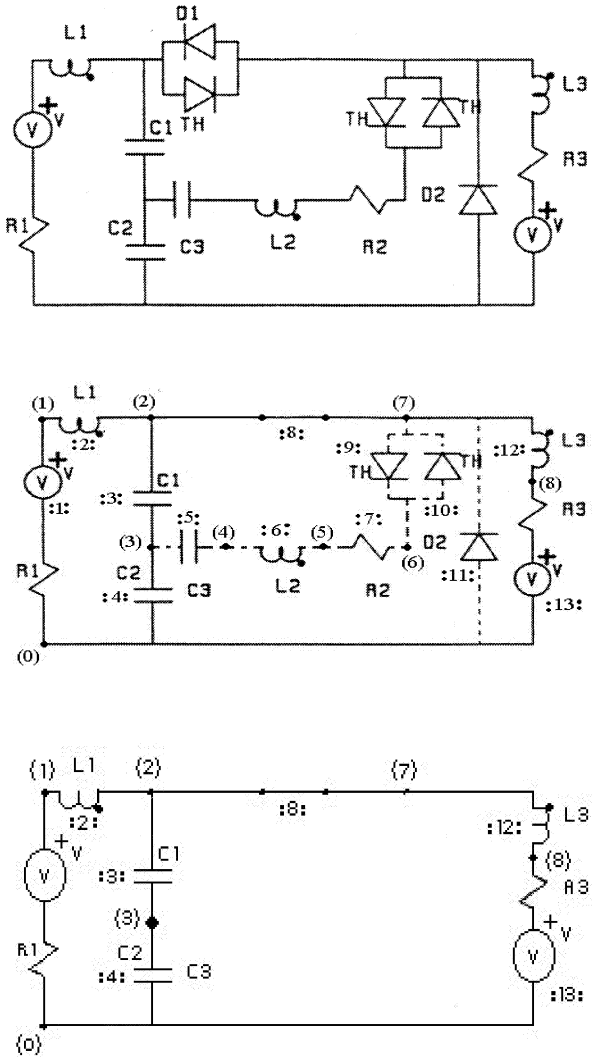


Figure: 2. The semi-conductors are simulated by Perfect switches.

When the semi-conductor is "on", we link the two nodes by a semi-conductor; when the semi-conductor is "off"; w e remove the link between the two nodes. At each step, the topology of the electric circuit is simplified compared to the original circuit. The number of differential equations is lower than the one obtained by the fixed topology simulation. The main difficulty of the general variable topology method: this method makes the knowledge of the voltage across a turned off semi-conductor difficult to obtain.

METHOD PROPOSED IN THIS STUDY

From the data (topology and components) and for each phase of operation, it is necessary to determine the minimal topology, the new value of the input nodes and output of the semi-conductors after removal of the branches traversed by a null current and the value of the voltage for each blocked semi-conductor.

Automatic removal of the blocked semi-conductors and pending node branches.

Consider a complete circuit containing a certain number of active and passive branches. For each blocked semi-conductor, one removes the corresponding branch. Then the input nodes and output are separated, which gives a circuit with two parts: a side node for the input, and another for the output. The main problems to be solved at this stage are:
1) Determination of the new numbers of input nodes and output at each blocked semi-conductor (and removed) when there is removal of branches in series, terminated by pending nodes.
2) Determination of the value of the terminal voltage of each blocked semi-conductor and removed. This value depends on the voltage of the removed capacitive branch or when it contains a voltage source.

Automatic determination of the new input nodes and output for each removed blocked semi-conductor.

This is dene in a first step analysis of the circuit branch by branch and by removed every branch containing a blocked semi-conductor. In a second step the process is repeated by checking if the output of esch branch is equal to the input node (e) of the blocked semi-conductor.
a) If so, we keep the number of this branch and assign a degree for the node (e) = degree of the node (e)+1.
b) If not, we verify if the input of this branch is equal to the node (e)
-If yes we keep the number of this branch and we assign a degree for the node (e) = degree of the node (e) +1.
- If not, we move to the next branch.
The same process is repeated for the output node (s).If the degree of the node (e) or the node (s) is equal to 1, the branch that connects this node is pending and this branch is removed. When the degree of the node is not equal to 1, this node is the new number for this blocked semi-conductor which is stored in the matrix G (X, Z).

Automatic determination of the voltages across blocked semi-conductors.

The objective is to determine the voltage across each blocked semi-conductor. It should be noted that the search for the new input and output nodes is not sometimes sufficient to know the variation of the voltage across the removed blocked semi-conductors for each branch containing a capacitor or a voltage source contributes to the determination of this voltage. For that, we will show how, starting from the removed passive branch we calculate the

value of the voltage across each blocked semi-conductor. We start by defining a function f (X, Y), which determines the terminal voltage of each blocked semi-conductor.

$$f(X, Y) = VC(X, Y) + VE(X, Y) \quad (1)$$

With:
X = number of branch containing a blocked semi-conductor,
Y = 1 or 2 according to whether one works on the corresponding output node or the input node of the blocked semi-conductor at each instant of time, we are able to define the values of the functions f (X, 1) and f (X, 2). These two functions allow is to know the voltage across each blocked semi-conductor by applying the following relation:

$$F(K)=(U(G(K, 2))+f(K, 2))-(U(G(K, 1))+f(K,1)) \quad (2)$$

With:
F (K): Voltage across the blocked semi-conductor numbered K. U (G (K, 2) + f (K, 2) Voltage of the input node (E) of the blocked semi-conductor compared to the reference node (O) U (G (K, 1)) + f (K, 1) Voltage of the output node (S) of the blocked semi-conductor compared to the reference node (O). It should be noted that at each step of calculation the value of the voltage of each blocked semi-conductor must be calculated. We study now the construction of matrices VC (X, Y) and VE (X, Y).

Case of the capacitive branches VC (X, Y)

Each capacitive branch connected to a node of degree 1 is removed. The value of the voltage accross these terminals is stored in VC(X,1) for output node of the blocked semi-conductor numbered X and in VC(X,2) for the input node of this blocked semi-conductor.
With:

$$VC(X, 1) = VC(X, 1) + F(K) \quad (3)$$

$$VC(X, 2) - VC(X, 2) + F(K) \quad (4)$$

During a whole phase of operation, the values of VC (X, 1) and VC (X, 2) remain constant. However, for each new phase of operation the matrix is initialized with zero value.

Case of the generators of voltage

The difference between a capacitive branch and a branch containing a generator of voltage comes owing to the fact that the voltage of a generator of voltage can vary as a function of time. Thus, its value is F (K) = VM (K) where VM (K) is the amplitude of the generator of voltage. It is thus necessary to recompute the voltage of the generator for each step of calculation and to store it in VE (X, Y).

AUTOMATIC DETERMINATION OF THE VALUE OF THE CURRENTS THAT CROSS EACH REMOVED CONDUCTING SEMI-CONDUCTOR

The current in a semi-conductor is the algebraic sum of the currents of the branches that connect the input node or the output node. It is to be noticed that the knowledge of the information stored in matrices MAENT (X, X1), MASORT (X, X1), SIGC (X, X1) enable us automatically to know the value of the currents which at every moment cross all the conducting semi-conductors of calculation (Boulos 20002).

FORMING AND SOLVING THE EQUATIONS

In order for the program to be automatic, it must establish the equations by itself. This is achieved in two steps:
The first step consists of the topological study of the circuit, and the second step consists of establishing the equations. An electronic circuit is represented by a group of branches connected together and the equations are established using mathematical technics. This enables the analysis of a great number of different circuits, for which the formation and solving of the equations would otherwise require long and prohibitory calculations. The topological data of the considered circuit as a directed graph is translated in two ways:
- Formation of subgraphs such as trees and coarbres
- Formation of the matrices (matrix incidence...).
We regard the circuit as a whole of directed branches connecting certain number of nodes, the whole constituting a circuit. We point out the necessary definitions:
- Branches: A branche is a segment, which connects two nodes. It is directed positively according to positive direction selected for the setup curent.
- Node: a node is a point where are connected two or several branches.
In a directed circuit formed of " n " nodes and "b" branches one defines the matrix incidence of A (n, b) in the following way.
A (n, b) = (Aij) such as:
Aij = 1 if the branche j starts from node i
Aij = -1 if the branche j arrives at node i
Aij = 0 if node i is not an end of the branche j
The dimension of this matrix is thus n*b. An branche starts from a node and arrives at an another node:
the terms of a column of the matrix (Aij) are all null except two which are equal 1 and -1. The number of nonnull terms of a line indicates, for each node, the arriving number of branches or on the basis of this node. There must be at least two nonnull terms, elze it means that there is an error in the representation of topology. One calls Ar the matrix of reduced incidence of dimension (n-1, b).

State variable method

The state variables are the currents in the inductances and the voltages across the capacitors. The knowledge of these values and of the current source or the voltage source enables to determine the functioning of the circuit, and to establish the two state equations:

$$DX/DT = A*X + B*U \quad (5)$$

$$Y = C*X + D*U \quad (6)$$

with Y = output equation. The vector X contains the voltages across the capacitors and the currents in the inductances. The vector U represents the input. In order to solve these equations mostly the exponential method is used or the method of Runge Kutta of order 2, 3 or 4. The stamp A, B, C and D are constant during the time interval separating two changes of conditions.

- Exponential method.
In the exponential method, we get the solution (Pelletier 1971):

$$X(t+DT) = \exp(A*DT)*X(t) + A^{-1}(\exp(A*DT-1)*B*U \quad (7)$$

DT is a calculation step.

- Runge Kutta method
The Runge Kutta method (Pelletier 1971):
DY/DT = f (X, Y)
Y (X0) = Y0
It is required to determine the value of the function at the point X (n+1), given (Xn, Yn) so that.

$$Y_{n+1} = Y_n + \sum_{i=0}^{n} A_i * K_i \quad (8)$$

$$where\ K_0 = DT* f(X_n * Y_n)$$

$$K_1 = DT* f(X_n + \mu_1 * DT, Y_n + (W_{1,0}) * K_0)$$

Also, the general equation:

$$K_i = DT* f(X_n + \mu_i * DT, Y_n + \sum_{j=0}^{i-1} W_{ij} * K_j)$$

For the application, we chose the setting in equation by the exponential method is used and the method of Runge Kutta of order 2, 3 or 4.

Runge Kutta Order 4.
Knowing Uc (i) and Il (i) has inastant it t (i) we can determiner has the moment t (i+1), IL (i+1) and Uc (i+1). We take Dt like step of calculation and we apply the following algorithms:

$$Il\ (i+1) = IL\ (i) + 1/6\ (T1 + 2T2 + 2T3 + T4) \quad (9)$$

$$Uc\ (i+1) = Uc\ (i) + 1/6\ (T11 + 2T22 + 2T33 + T44) \quad (10)$$

With Tj, Tjj defined by the equations (Pelletier 1971).

Runge Kutta Order 3.
The same method of application that Runge Kutta order 4 but we take the following algorithms:

$$Il\ (i+1) = IL\ (i) + 1/6\ (T1 + 4T2 + T3) \quad (11)$$

$$Uc\ (i+1) = Uc\ (i) + 1/6\ (T11 + 4T22 + T3) \quad (12)$$

Runge Kutta Order 2.
The same method of application that Runge Kutta order 4 and 3 but we take the following algorithms:

$$Il\ (i+1) = IL\ (i) + T2 \quad (13)$$

$$Uc\ (i+1) = Uc\ (i) + T22 \quad (14)$$

CHOICE OF THE STEP OF CALCULATION AND ANALYSIS BY THE STEP VARIABLE METHOD

The difficulty of is method of calculation is the presence of inductive and capacitive nonlinear elements. The modeling of inductances and the capacitors imply a linear variation of current and tension during very short a Dt time. The convergence due to the calculation step DT poses no problem with the exponential method, but it does in the iterative method Runge Kutta where the approximate formula connot be applied, if DT is not chosen small enough, it causes false results. If DT is chosen too small, the calculation time gets too long. The state variable method, for each change of the operating mode, the matrix must be constructed fromthe equation, and then the equations are solved. At each step we have only to calculate the state variables. By determining the turn-on or turn-off operating mode, the switching voltage and current can be calculated at each step of integration. In order to reduce the computing time, whw have to make a compromis on the step. Then each time the mode changes we return to the last state and integrate with a smaller step. Thus the integration step is reduced considerably and hence the total integration time is minimised. The step choice depends on the choice of the solution method. The state variable method and the solving by exponential matrix is more appropriate than the other methods but too prohibitory.
For example; an the integration step the Runge Kutta method is determined as a function of the time constants (RC or LC), the period T and the amortization factor M.
- In case circuit LC the time step should be smaller than the period $T = 2*\pi\sqrt{L*C}$.
- In case circuit RC the time step should be smaller than the time constant R C.
- In case circuit RLC the time step is a function of the amortizing factor M:
- If M is big, then the step is the same as in the case of the RC circuit.
- If M is small, then the step is the same as in the case of thee LC circuit.
This is why we have to compromise between the choice of thee time step value and the accuracy of the zero value.

Choice of a constant step

The step is selected from the beginning of simulation. It is equal to the 1/100 of the smallest time-constant of complete topology.

Choice of a variable step

The method of the variable step consists in binding the step of calculation to the change of configuration. For each change of configuration the complete topology of the network is replaced by a minimal topology. The step of calculation is equal to the 1/100 of the smallest time-constant of this minimal topology.

The choice of the method with variable topology makes it possible automatically to adapt the step of calculation to each change of topology to obtain a precise result with a saving of time of simulation.

EXAMPLE OF APPLICATION AND RESULTS OBTAINED

In order to validate the choice between the three algorithms of Runge Kutta of order 4,3 and 2 we took as example of application the calculation of the current and the tension (Il and Uc in a filter RLC). As an exemples use as filters of input of the chopper figure 3 which uses the method with variable topology. We studied the percentage of error (Err %) according to the step of calculation DT that varies from 10^{-7} to 10^{-5} for the three algorithms of Runge Kutta of order 4,3 and 2 with:

Err % =((true value - calculated value) / calculated value)

The true value is calculated by applying the solution of the exponential method of matrix.

Indeed, the resolution by the exponential methode of matrix does not pose a problem of convergence with the choice of the step of calculation DT. The calculated value is that found by using one of the three algorithms of Runge Kutta of order 4,3 and 2. We will find the results in the graph figure 4 and figure 5 which represent the values of Err % Uc and of Err % Il according to DT. In our example with filter RLC used, the damping ratio M is about $18.6.10^{-4}$. M is too smaller then DT wich corresponds to the case of circuit LC. Therefore DT must be negligible in front of the period T $=2*\pi\sqrt{L*C}=8.43.10^{-4}$. We choose DT less or equal to 2.10^{-6} to obtain a more precise calculation. We realize that by accepting Err % inferior or equal to 2 %, the best step of calculation for Err % Uc and Err % Il will be DT equal to 2.10^{-6}. Therefore for a step of about 1/2000 of period T, the difference in percentage of error between Runge Kutta 4,3 and 2 will be about 0.035 %. This margin of error is negligible compared to the calculated values. However the difference in computing time between methods Runge Kutta 3 and 4 and method Runge Kutta 2 is 35 %. If we take the method of exponential of matrix, we observe that the computing time is the double of that of the algorithm of Runge Kutta order 2. It is thus preferable to use the method of Runge Kutta of order 2. With R = 0.05 Ω, L = 18 μ H and C = 1000 μ F.

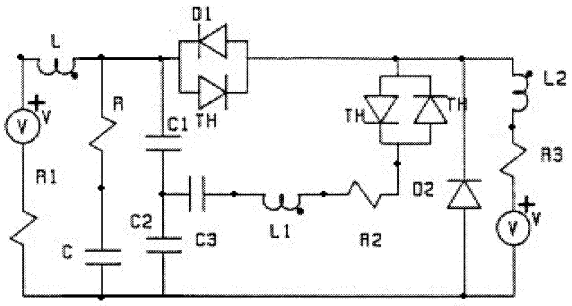


Figure 3: Schema of the simulated chopper.

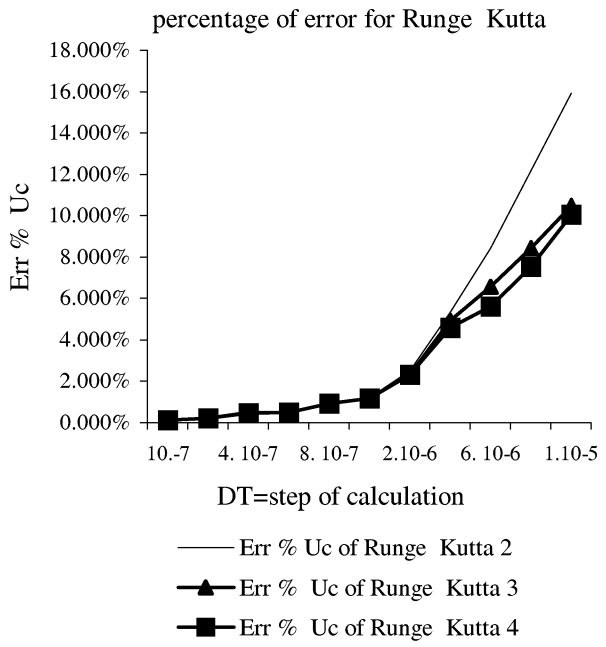


Figure 4: Represent the percentage of error Err % Uc for Runge Kutta ordre 2,3 and 4.

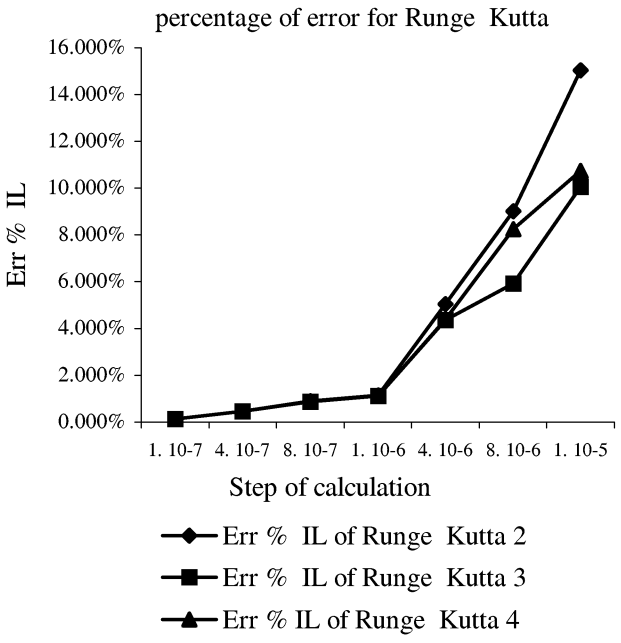


Figure 5: Represent the percentage of error Err % Il for Runge Kutta ordre 2,3 and 4.

RESULTS

In order to illustrate the mentioned methods, the results obtained by simulating a chopper figure 3 are shown. Table I shows the time ratios obtained when using the example of the chopper figure 3. The gain is between 2 and 3.

$\frac{\overline{W}()}{W()}$	$\delta_n\%$	$\delta_t\ \mu_s$	$\mu=\frac{\delta t}{TT}\%$
20 / 72	27%	$95\mu_s$	47,5 %
42 / 72	58%	$75\mu_s$	37.5%
52 / 72	77%	$30\mu_s$	15 %

Table I: The time ratios obtained.

Where:

$\overline{W}$ (n-1,n): calculating matrix corresponding to the minimal topology.

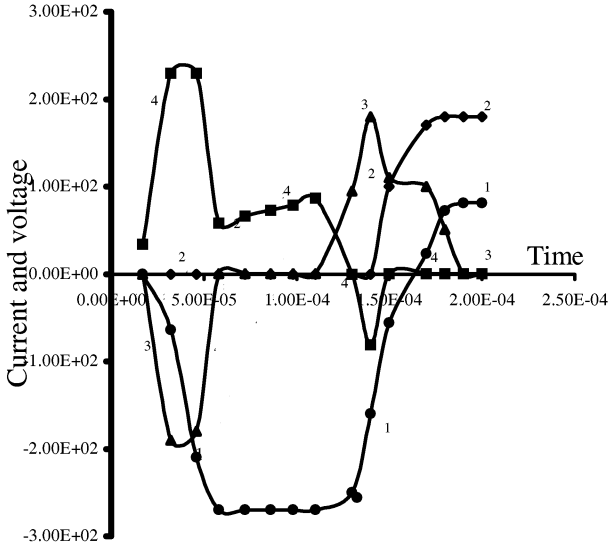
W(n-1,n): calculating matrix corresponding to the fixed topology.

$\delta_n\%$: Percentage of the number of elements used for the variable topology.

δ_t: Simulation time for $\overline{W}$ (n-1,n) during a certain simulation period.

μ: Percentage of the simulation time.

TT: Simulation period.



1- Voltage across a condensator

2- Voltage across a diod D1 and the antiparallel TH

3- Current in a condensator C3

4- Current in a diod D1 and the antiparallel TH

Figure. 6 Represents the results of the simulation using the variable topology.

We present the form of the currents and the tensions for the capacitive branch C3 and the diode D1 and the antiparallel thyristor TH.

CONCLUSION.

The aim of this paper is to study and realization of a general program of numerical simulation of static inverters by the method of variable topology, which permits:

* To produce the minimal topology and to generate automatically the equations at each configuration change.
* To reduce the order of the system to be resolved, allowing important cuts in computational time overhead.
* The problem of varying time constants in the solution with the state variable method in the semi-conductor circuits is eliminated by removing the open circuit branches.
* Better conditioning of the resistive matrices because of the disappearance of the important reports/ratios of resistances (Ron/Roff).
* More precise results simulation because the secondary time constants are avoided.
* The choice of a variable step of calculation makes it possible to have a greater speed of simulation and more precision by a determination of the smallest time-constant than we will choose among time-constants of all the under-circuits which form minimal topology.
* It proves that the method of Runge Kutta order 2 is fastest for a more precise result.

REFERENCE

Antognetti P., G. Massobrio 1988. 'Semi-conductor Device Modeling with SPICE" *McGraw-Hill Book.*

Boulos F. 1988. "Etude et réalisation d'un programme de simulation de convertisseurs statiques. Utilisation d'une méthode à topologie variable". *Thèse de docteur de l'Université Paris VI.*

Boulos F. 2001. " Numerical simulation of power electronic Circuit. Use of variable topology method. Modelling of the Semi-conductor as parfect switches". *SCS MESM 3rd Middle East Symposium on Simulation and Modelling,* (September 3-5), PP. 97-100.

Boulos F. 2002. " Calculating the current in conducting semi-conductors modelled by perfect switches". *SCS MESM, 4th Middle East Symposium on Simulation and Modelling,* (September 28-30), PP. 217-222.

Eisenar H. and Hofmenister H 1972. 'Digitale nachbildung von electrischen netz werken mit dioden und thyristoren "*archin fur Electrotechnik* n55.

Lakatos L. 1979. " A New method for simulating power semi-conductor circuits". *IEEE Trans. Ind. Electro.* Vont Inst. Vol IECI - 26 n 1 (Feb).

Nagel L. N., Pederson D-O.1973. *"SPICE"* Simulation program with integrated circuit emphasis" *Berkeley California: University of California Electronics Research laboratory* Memorandum ERL,M 382 (Apr 12).

Pelletier P. 1971. 'Techniques nu mériques appliquées au calcul scientifique". *Masson.*

Rajagopalan 1978. 'Simulateur digital des convertisseurs de puissance à thyristors" *Can. Elec. Eng. J.* n 1PP 5-10(Jan).

Schonek J. 1977 Simulation numerique des convertisseurs statiques application a la conception et a l'optimisation des convertisseurs. *These de docteur-Ingenieur, INP Toulouse.*

BIOGRAPHY OF AUTHOR

FAOUZI BOULOS was born in Byblos in Lebanon. He studied in France at the university of Pierre and Marie Curie (Paris VI). He obtained a thesis in electronics in 1988. He worked a few years in France and returned to Lebanon. Since 1995, he is professor at the department of physics of Lebanese University of sciences II. His e-mail address is: faboulos@ul.edu.lb

INFORMATION THEORETIC APPROXIMATIONS FOR THE M/G/1 RETRIAL QUEUE WITH UNRELIABLE SERVER

Aissani A., Smail R.
University of Science & Technology (USTHB),
BP 32 El Alia, Bab-Ez Zouar, 16111,
Algeria.

E-mail: aaissani@wissal.dz

KEYWORDS

Retrial queues, Breakdowns, Information theoretic approximation, Entropy

ABSTRACT

This talk focuses on M/G/1 retrial queue when the server is subject to random breakdowns. The theory of retrial queues developed last years provides a conventional framework for solving such models. Performance measures for such systems are available in explicit form. Here, we give more simple approximations obtained via the maximum entropy formalism. These approximations are obtained according to the available information about the service time, repair time probability distributions and the steady-state distribution of the system state.

1.INTRODUCTION

Retrial queues (see Falin and Templeton 1997) have been used in many concrete applications as the modelling of Digital Cellular Mobile Networks (Sun Jong Kwon et al 2001), Local Area Networks with star topology (Janssens 1997) and so on. However all the models used there, neglected breakdown process.

Here, we consider an M/G/1 retrial queue when the server is subject to random breakdowns. This model is well known and its performance measures are available in explicit form (Kulkarni and Choi 1990; Aissani and Artalejo 1998). This model has been studied in the past by analytical methods (imbedded Markov chain, supplementary variable, diffusion approximation…). The interested reader can refer also to some related topics in (Wang and Cao 2001; Artalejo and Gomez-Coral 1998; Djellab 2002). The solutions are obtained in terms of generating and Laplace transforms, so it is interesting to obtain more simple estimations. In this paper, we present information theoretic approximations for this model obtained according to the available information about the service time and repair time probability distributions.

The next section is meant for description of the model. Section 3 presents a succinct presentation of the maximum entropy formalism. Section 4 concerns itself with first order approximation for which an explicit solution can be derived. In section 5 we discuss the second order approximation for which additional information is available about second order moments. Finally, we conclude the study by some numerical examples showing the quality of the obtained approximations.

2. THE M/G/1 RETRIAL QUEUE WITH UNRELIABLE SERVER

Consider an unreliable M/G/1 retrial queue: customers arrive according to a Poisson process with rate $\lambda>0$. The server is subject to two types of Poisson interruptions (breakdowns) with rates η and μ respectively. Independent breakdowns may occur when the server is idle, while active breakdowns occur when the server is busy by the service of a certain customer. The service times, and repair times assumed to be mutually independent random variables with arbitrary distributions. A customer who finds the server busy or out of order leaves the system and repeat his request after an exponentially distributed time with intensity θ. Customers whose service is interrupted by an active breakdown must decide either to join the retrial group (with probability c) or leave the system.

The considered system is described by a discrete stochastic process $\{Q(t), X(t)\}$ on the state space $\{0,1,2,3\}\times IN$, where $Q(t)$ is the number of customers in the retrial group at time t and

$X(t)=0$ if the server is operative and idle at time t
$=1$ if the server is operative and busy
$=2$ if the server is down due to an independent breakdown
$=3$ if the server is down due to an active breakdown.

From (Kulkarni and Choi 1990) it is well known that the steady state probability that the system is in state (i,j).

$$P_{ij}=\lim_{t\to\infty}P\left(X(t)=i,Q(t)=j\right),$$

$i\in\{0,1,2,3\}$,$j\in\{0,1,2,\ldots\}$ exists if

$$\rho=\lambda\frac{1-\tilde{\tilde{F}}(\mu)}{\mu}\left(1+\mu\left(E(D_i)+\frac{c}{\mu}\right)\right)<1,$$

where $\tilde{\tilde{F}}(s)$ is the Laplace-Stieltjes transform of the service time distribution and D_i is the (random) duration of an independent breakdown.

3. MAXIMUM ENTROPY FORMALISM

Let Q be a system with discrete state space $S=\{S_n\}$ such that the available information about Q places a number of constraints on the probability distribution $p=\{p(S_n)\}$. In general we can assume that the constraints concern mean values of suitable functions $\{f_i(S_n)\}_{I=1,m}$, where $m<card(S)$. The principle of maximum entropy (PME) (Kouvatsos and Tabet-Aouel 1989) states that, of all distributions which satisfy the constraints, the minimally biaised distribution is the one which maximises the system's entropy function

$$H(p)=-\sum_{S_n\in S}p(S_n)\log\, p(S_n) \qquad (1)$$

subject to the constraints

$$\sum_{S_n\in S}p(S_n)=1 \qquad (2)$$

$$\sum_{S_n\in S}f_k(S_n)p(S_n)=<f_k>,k=1,\ldots,m \qquad (3)$$

where $\{<f_k>\}$ are the prescribed mean values defined on the set of functions $\{f_i(S_n)\}_{I=1,m}$. The maximisation of (3.1) subject to the constraints (3.2), and (3.3) can be carried out using Lagrange's method of undetermined multipliers and leads to a solution of the form

$$p(S_n)=\frac{1}{Z}\exp\left(-\sum_{k=1}^{m}\beta\, f_k(S_n)\right) \qquad (4)$$

where $\{\beta_k\}$ are the Lagrangian multipliers determined from the set of constraints (3.4). The function Z is known on statistical physics as the "partition function" , and is given by

$$Z=\exp(\beta_0)=\sum_{S_N\in S}\exp\left(-\sum_{k=1}^{m}\beta\, f_k(S_n)\right) \qquad (5)$$

where β_0 is the Lagrangian multiplier determined by the normalisation constraint (3.2).

4. FIRST ORDER APPROXIMATION

Return now to our M/G/1 retrial queue with unreliable server described in section 2. In the work of (Aissani and Artalejo 1998)] it is provided explicit formula for the partial moments $M_i^k=\sum_{j=0}^{\infty}j^kP_{ij}$, which can be used as a priori information. If the only available information is given by M_i^0, M_i^1, i=1,2,3, then according to the PME, the probability distribution of system state can be obtained under an explicit form and it is given under the following geometric form

$$\hat{P}_{in}=\alpha_i\beta_i^{\,n}\ ,\ i=0,1,2,3$$

where the constant α_i, β_I can be evaluated in an explicit form. Their evaluation needs only simple arithmetion operations.

5. SECOND ORDER APPROXIMATION

In this section, we solve the problem by adding additional information about the second order partial moments. The explicit form of these second order moments are also given in (Aissani and Artalejo 1998). In this case, it is not possible to obtain an explicit solution of Lagrange's parameters. However, we can obtain a numerical solution. In our case, the maximisation of the entropy function $H(P_i)=-\sum_{n=0}^{\infty}P_{in}\log\left(P_{in}\right)$, $i\in\{0,1,2,3\}$ subject to the constraints about the partial moments M_k^i, $i\in\{0,1,2,3\}$, $k\in\{0,1,2\}$, leads to a solution of the form

$$\hat{P}_{in}=Z_i^{-1}\exp\left(-\beta_1^i n-\beta_2^i n^2\right)$$

$$Z_i=\frac{1}{M_i^0}\sum_{n=0}^{\infty}\exp\left(-\beta_1^i n-\beta_2^i n^2\right)$$

where β_1^i and β_2^i are the Lagrange's multipliers corresponding to the constraints. Using standard methods of numerical analysis the computation of the parameters $\beta^i{}_1$and β_2^i, is reduced to find minimum of the convex potential

function

$$F_i(\beta_1^i,\beta_2^i)=\log\sum_{k=0}^{N}\exp\left\{-\sum_{J=1}^{2}\beta_j^i\left(k^j-\frac{M_i^j}{M_i^0}\right)\right\},$$

$i\in\{0,1,2,3\}$ and N is the orbit truncated size.

6. NUMERICAL TESTS

In this section we show some comparisons between maximum entropy approximations and exact values of the unreliable $M/H_2/1$ retrial queue with Hyperexponential (H_2) repair time distribution. The used statistics are indicated in table 1. From this table, we observe that first order approximation is decreasing. All experiments show that performance of approximation is improved by additional information given by second order moments. However, the performance of this second order approximation gets worse when retrial rate decreases. This is also indicated in last line of the table in which we have computed Shannon's entropy. Clearly, the entropy decreases with number of moments used as a priori information. The difference in computation of Shannon's entropy is due to truncation used for the second order approximation.

7.CONCLUSION

We have presented information theoretic approximations of system state distribution in an M/G/1 retrial queue with unreliable server. This approach leads to simplifications and enough accurate approximations for practical purpose. This approach can also be used for other performance measures (waiting time,…).

8.REFERENCES

Aissani, A. and Artalejo J.R. 1998, "On the single server retrial queue subject to breakdowns", *Queueing Syst.* 30, 309-321.

Artalejo, J.R. 1999, "A classified bibliography of research on retrial queues: Progress in 1990-1999", *TOP,* vol. 7, N°2, pp.187-211.

Artalejo, J..R. and Gomez-Corral, A. 1998, "Unreliable retrial queues due to service interruptions arising from facsimile networks", *Belgian J. of Stat., Operat. research and Computer Sci.* 38:31-41,1998.

Djellab, N.V., 2002. "On the M/G/1 Retrial Queue subjected to breakdowns" *RAIRO Operations Research* 36, 299-310.

Falin, G..I..;Martin Diaz, M. and Artalejo, J..R., 1994. "Information theoretic approximations for the M/G/1 retrial queue", *Acta Informatica*, 559-571, 1994..

Falin, G.I. and Templeton, J.G.C., 1997, "*Retrial queues*", Chapman and Hall, London.

Kouvatsos, D.D. and Tabet-Aouel, N. 1989."A maximum entropy priority approximation for a stable G/G/1 queue", *Acta Informatica* 27, 247-286.

Kulkarni ,V.G. and Choi B..D. 1990." Retrial queue with server subject to breakdown and repairs", *Queueing systems*, 7, 191-208.

Janssens, G. K. 1997, "The quasi-randum input queueing system with repeated attempts as a model for a collision-avoidance star local area network", *IEEE Trans. on Commun*; vol. 45, N°3, 361-364.

Sun Jong Kwon et al, 2001. "Performance analysis of CDPD Sleep Mode for power conservation in Mobile End Systems", *IEIC Trans. Commun.,* vol. E84,N°10.

Wang J. and Cao J. 2001, "Reliability analysis of the retrial queue with server breakdowns and repairs", *Queueing Systems*, 38, 363-380.

Table 1. Comparison of exact and approximate solutions for an $M/H_2/1$ retrial queue with unreliable server: $\lambda=1$, $E(S)=0.25$, $\theta=4,\eta=0.5,\mu=2,E(D_i)=E(D_b)=0.25$, $c=0.8$.

Hyperexponential repair time distribution $f(x)=0.5^{e-2x}+4.5e^{-6x}$												
n	P_{0n}	P_{1n}	P_{2n}	P_{3n}	$\hat{P}^{1}_{0n}$	$\hat{P}^{1}_{1n}$	$\hat{P}^{1}_{2n}$	$\hat{P}^{1}_{3n}$	$\hat{P}^{2}_{0n}$	$\hat{P}^{2}_{1n}$	$\hat{P}^{2}_{2n}$	$\hat{P}^{2}_{3n}$
0	0.5195	0.1179	0.0494	0.0090	0.5473	0.1068	0.0534	0.0318	0.5502	0.1073	0.0538	0.0165
1	0.0912	0.0574	0.0198	0.0423	0.0870	0.0429	0.0183	0.0205	0.0837	0.0426	0.0180	0.0237
2	0.0264	0.0258	0.0074	0.0291	0.0138	0.0172	0.0063	0.0132	0.0138	0.0170	0.0062	0.0233
3	0.0086	0.0115	0.0027	0.0157	0.0022	0.0069	0.0022	0.0085	0.0025	0.0069	0.0022	0.0157
4	0.0030	0.0050	0.0010	0.0078	0.0003	0.0028	0.0007	0.0055	0.0005	0.0028	0.0008	0.0072
5	0.0011	0.0022	0.0004	0.0037	0.0001	0.0011	0.0003	0.0035	0.0001	0.0012	0.0003	0.0023
6	0.0004	0.0009	0.0001	0.0017	0.0000	0.0004	0.0001	0.0023	0.0000	0.0005	0.0001	0.0019
Shannon's entropy			1.7558			1.7889				1.7712		

FLUID FLOW MODELLING SIMULATION

SIMULATION OF ELECTRIC FIELDS AROUND TRANSMISSION LINES FOR PROXIMITY PREDICTION

Amruth Sivalenka
Department of Industrial Engineering
The State University of New York
Buffalo, NY 14260
E-mail: avs@eng.buffalo.edu

Mohammed Safiuddin
Department of Electrical Engineering
The State University of New York
Buffalo, NY 14260
E-mail: safium@eng.buffalo.edu

Satish Mohan
Department of Civil, Structural, and Environmental Engineering
The State University of New York
Buffalo, NY 14260
E-mail: smohan@eng.buffalo.edu

KEYWORDS
Transmission Lines, Crane Safety, Electrocution, Proximity Prediction, Finite Element Modeling, Simulation.

ABSTRACT

Transmission lines are used in a power grid to carry electrical energy across the nation. Accidents due to electrocutions by direct physical contact of mobile cranes or other boomed vehicles moving in the vicinity of these transmission lines cost the nation heavy losses in the form of lives, workdays, and finances. A total of 5,348 workers were electrocuted from 1980 through 1992 in the USA, 411 workers electrocuted per year, an average annual rate of 0.4 per 100,000 workers. The year 1999 cost of these accidents was estimated at $259 million. 39% of these electrocution accidents involved boomed vehicles such as cranes. A potential approach in predicting the proximity of transmission lines to avoid electrocution is to analyze the electric fields created by the transmission lines in their vicinity. In this paper, we simulated the electric fields created by three most commonly used 3-phase transmission line configurations over long radial distances using finite element analysis. Simulations indicate that even at a radial distance of 40 meters away from a double-circuit transmission line configuration energized to 345 kV, the value of electric field due to it is about 150 Volts/meter, which can be detected by sensing devices to warn the mobile equipment of its close proximity to high voltage transmission lines.

1. INTRODUCTION

Electrical energy is transmitted across the nation through power grid. One of the critical elements of a power grid is the transmission line shown in Figure 1. During power transmission, electrical energy is transmitted from point to point at very high voltages to minimize transmission losses. These voltages can be as high as 1000 kV (Tamaki and Yoshibumi 1996). Almost all workers in the industrialized nations are exposed to energized transmission lines at some time during their work. In spite of all physical safety guidelines, working around these high voltage transmission lines can cause electrocution fatalities and serious injuries. In fact, electrocution is the second leading cause of death among construction workers (Hicks 2000).

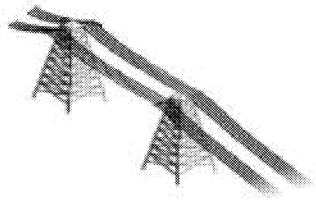


Figure 1: A Section of Power Grid

National Institute for Occupational Safety and Health (NIOSH) has identified the five case scenarios given in Table 1 that describes the incidents resulting in electrocutions. A total of 5,348 workers were electrocuted from 1980 through 1992 in the USA, 411 workers electrocuted per year, an average annual rate of 0.4 per 100,000 workers (Kisner and Casini 1998). In 1996, 116 electrocutions resulted from contact with overhead transmission lines (Occupational Safety and Health Administration 2003). In 1998 and 1999 alone, 277 workers died due to contact with transmission lines (Department of Labor 2000). On an average, seven workers are electrocuted every year in aerial lifts during tree trimming (Mohan 2000). In addition, metal and conductive scaffolds killed 47 workers during 1991 in similar accidents. The year 1999 cost of these accidents was estimated at $259 million (Mohan 2000).

Table 1: Percentages of Crane-Related Electrocutions

Cause	Percentage
Direct worker contact with an energized transmission line	28 %
Direct worker contact with energized equipment	21%
Boomed vehicle contact with an energized transmission line	18%
Improperly insulated or damaged equipment	17%
Conductive equipment contact with an energized transmission line	16%

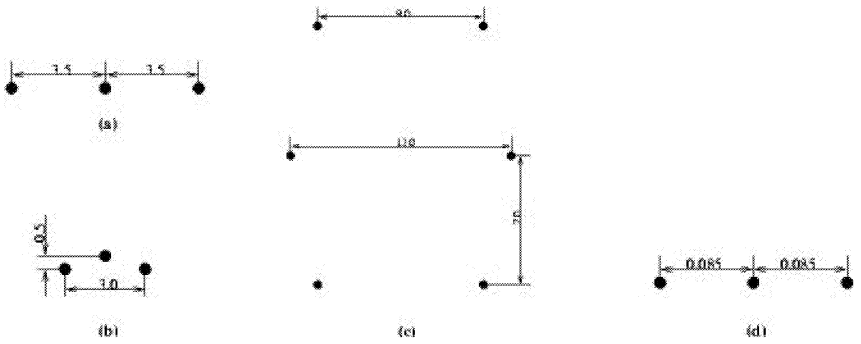


Figure 2: Sectional Views of (a) Horizontal Configuration, (b) Triangular Configuration, (c) Double-Circuit Configuration, and (d) Laboratory Model of Horizontal Configuration (All dimensions are in m, not to scale)

Federal regulations (Code of Federal Regulations 1926) mandate crane operators to maintain safe distances of separation from transmission lines. Current industry practices employ visual inspection and manual signaling to avoid direct physical contact with transmission lines. However, tall and mobile boomed equipment such as a crane can close quickly the safe distance separating it from overhead transmission lines because of optical illusion in inspection, delays in manual signaling, or lack of perfect co-ordination between the equipment operator and the guide. Our aim is to mitigate the loss due to accidents involving transmission lines by developing a sensor system that can detect automatically the proximity of transmission lines with respect to the mobile equipment and trigger alarms, eventually deactivating the equipment, if any part of the equipment gets into the hazard zone. By laws of physics, transmission lines create electromagnetic fields around them. However, transmission line proximity prediction by sensing these fields is possible only by analyzing the electric rather than magnetic fields created by them because of the following fact: The magnetic field created by a transmission line depends on the current flowing through it while the electric field created by it depends purely on its energized voltage, which remains constant for all practical purposes (Nguyen et al. 1996). However, the major disadvantage with electric fields is that they are affected by nearby structures, crane equipment, ambient electromagnetic noise, and fluctuation in voltages and geometry of transmission lines. In our current effort to develop a sensor system that can predict transmission line proximity for any possible transmission line configuration and operating voltage, we are designing and developing conductor-based sensor heads that detect electric fields. The electric field values sensed by such heads placed at various locations on the mobile equipment can be used computationally by a sensor system to predict the proximity of transmission lines. The fundamental step in the effective and universal design of such a sensor system is determining and understanding the complex patterns of electric fields around all possible transmission line configurations over a wide range of operating voltages and radial distances. However, conducting actual site experiments and measuring electric field values to determine the patterns of electric field for an exhaustive range of transmission line configurations, operating voltages, and radial distances is impractical and not cost-effective. An alternate yet accurate approach to determine the desired patterns is to simulate electric fields using finite element analysis. In this paper, we present the results of finite element simulation of electric fields created by three commonly used 3-phase transmission line configurations, for a voltage range of 46-345 kV, up to a radial distance of about 40 m: Horizontal, Triangular, and Double-Circuit.

The rest of the paper is organized as follows. In section 2, we describe the assumptions and methodology of our simulation. In section 3, we present the results and describe the interpretation of them. Finally, in section 4, we conclude the study with a note on future work.

2. FINITE ELEMENT SIMULATION

Three commonly used transmission line configurations are horizontal, triangular, and double-circuit configurations (Mohawk 1990). Any other transmission line configuration is a combination of these three configurations and its electric field pattern can be obtained by superposing properly a combination of electric field patterns of these three configurations. Hence, we restricted our simulations to these three configurations. We used the specific geometries depicted in Figures 2 (a), (b), and (c) for simulation purposes. Two crucial assumptions of our simulation are:

1. Transmission lines are infinitely long conductors with negligible diameter.

2. Effects of nearby structures, crane equipment, ambient electromagnetic noise, and fluctuation in voltages and geometry of transmission lines are negligible.

We developed the two-dimensional finite element model shown in Figure 3 (a) consisting of uniformly distributed

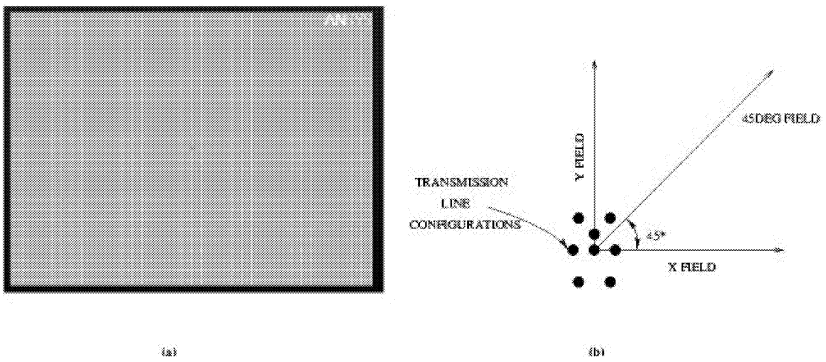


Figure 3: (a) A Zoomed View of the Uniform Model. (b) Electric Field Decay Directions: X FIELD, Y FIELD, and 45DEG FIELD

10,000 eight-node quadrilateral elements using ANSYS Multiphysics 5.7 (ANSYS 2000) as the simulation bed. In the process, we modeled a square with a side length of 100 m and meshed it by setting the element edge length to 1 m. As a result, each element of the model took the shape of a square with four of its nodes at its four vertices and the other four at the mid-points of its sides. This entailed the model to take the form of a square grid of nodes or mid-nodes every 0.5 m. We used the created model consistently for all of our simulations to eliminate the effects of unequal inaccuracies in different models. Our idea was to pick up nodes at proper locations of this model and denote them as transmission lines since we assumed their diameters negligible. To test the validity of our assumptions and methodology of simulation, we set up a laboratory model of horizontal transmission line configuration in an experiment as shown in Figure 2 (d) and maintained it at 3-phase 4 kV. We insulated its surroundings electrically to minimize the ambient electromagnetic noise and measured the values of electric field (M FIELD) at various radial distances for a single instant of time using a sensor head (Mohan et al. 2003). We simulated the laboratory model with our simulation bed with the same assumptions and methodology and compared the simulated results with the measured values of electric field. Once convinced that our assumptions and modeling theory are correct, we simulated the electric field patterns of 3-phase 46 kV horizontal, 115 kV triangular, and 345 kV double-circuit configurations and plotted the simulated results for one instant of time. We arrived at the choice of one instant of time because we analyzed that the electric field values sensed by our sensor head will follow trends similar to readings taken at any instant of time rather than specifically root mean square value or average value over the entire time period.

3. RESULTS

Since the decay of electrical field may not be same radially all around the transmission lines, we considered three important directions: one along the X-axis, second along the Y-axis, and the other along a 45^0 line in between the X- and Y-axes as shown in Figure 3 (b). We denote the magnitude of the total electric field on the lines along these directions as X FIELD, Y FIELD, and 45DEG FIELD respectively. A comparison of laboratory model's measured electric field (M FIELD) and simulated electric field (45DEG FIELD) is given in Figure 4. The values of experimentally measured electric field have been divided by a factor of 10 to accommodate the proportionality constant of the sensor head. It is evident from the graph that they follow the same trend line, thus validating the assumptions and methodology of our simulation. Graphs of decay of the electric fields of 3-phase 46 kV horizontal, 115 kV triangular, and 345 kV double-circuit configurations along the described directions are given in Figures 5, 6, and 7 respectively. (Numerical results of these graphs are available with the authors).

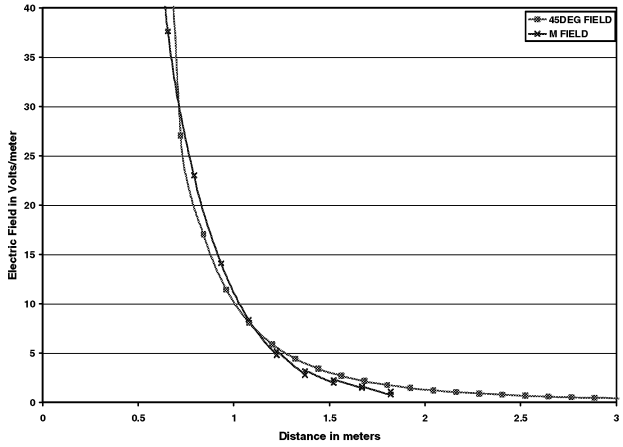


Figure 4: Radial Decay of Simulated and Observed Electric Fields around 3-Phase 4 kV Horizontal Line Configuration

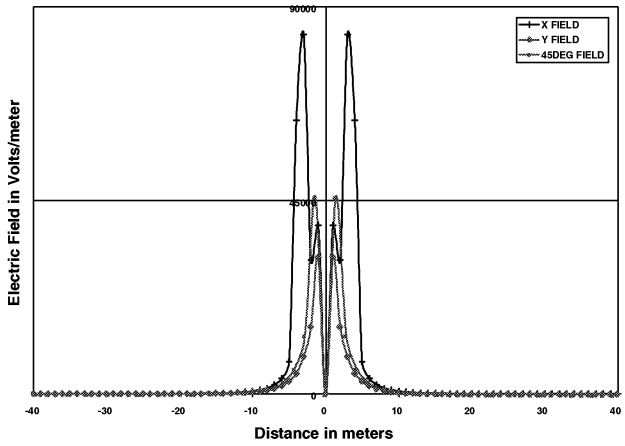


Figure 5: Radial Decay of Simulated Electric Field around 3-Phase 46 kV Horizontal Line Configuration

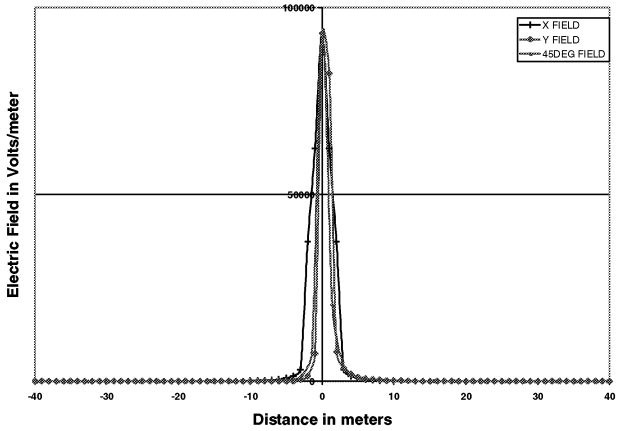


Figure 6: Radial Decay of Simulated Electric Field around 3-Phase 115 kV Triangular Line Configuration

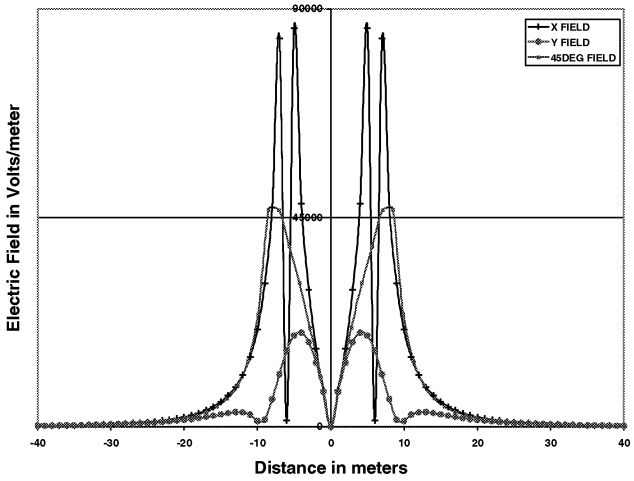


Figure 7: Radial Decay of Simulated Electric Field around 3-Phase 345 kV Double-Circuit Line Configuration

From results, we observed that a given configuration maintained at two different voltages has values of electric field differing by the factor of ratio of corresponding voltages at all corresponding radial distances away from the transmission lines. Though a comparison of the electric fields across various configurations simulated at same voltage appears fair, we chose to simulate them maintaining at different voltages because not all configurations are ever maintained at same voltages in practice. Moreover, since decay patterns for two different voltages of the same configuration differ by a factor, decay trends across the configurations can still be compared even though their voltages are different. Importantly, we are interested in decay patterns of certain commonly used configurations and their corresponding voltages for developing a sensor system to predict their proximity.

It can be observed from the graphs that the electric field decreases drastically with the increasing radial distance and dropped down by more than 90% of its peak value within 15 m of radial distance away from the lines. However, it is important to consider the movement of mobile equipment towards the transmission lines rather than away from them. Hence, the value of electric field increases drastically as the mobile equipment moves towards the transmission lines. This means that the sensor system should have very quick response time to detect the proximity of transmission lines and avoid direct physical contact with them. Another interesting observation is the behavior of transmission lines as a single line beyond a certain threshold radial distance. This can be seen as all the three fields X FIELD, Y FIELD, and 45DEG FIELD merge onto the same trend line at a certain radial distance away from the lines. These threshold values are approximately 9, 7, and 22 m for the simulated horizontal, triangular, and double-circuit configurations respectively.

4. CONCLUSIONS

We simulated the commonly used horizontal, triangular, and double-circuit line configurations and plotted the patterns of electric fields around them. These finite element simulation results match the field measurements. Simulated results indicate that electric field value even at a distance of 40 meters away from a double-circuit transmission line configuration energized to 345 kV, is high-enough to be picked up by sensors. Hence, simulated results are extremely useful for hazard assessment of transmission lines and designing safety products for workers in the vicinity of transmission lines. This study can be extended by incorporating the effects of the presence of nearby structures, crane equipment, ambient electromagnetic noise, and fluctuation in voltages and geometry of transmission lines. Simulating electric fields over a period of time using harmonic analysis is also an interesting extension to this study.

ACKNOWLEDGEMENT

This research was fully supported by National Institute for Occupational Safety and Health (NIOSH), US Department of Health and Human Services through grant numbered 5 R01-OH03803-03. The patenting process for the described sensor system is in progress.

REFERENCES:

ANSYS. 2000. "ANSYS 5.7 Manuals and Guides." *ANSYS Inc.* Southpointe, 275 Technology Drive, Canonsburg, PA.

Code of Federal Regulations. 1926. ``Cranes, Derricks, Hoists, Elevators, and Conveyors." A-15, Subpart N, *Title 29*, 1926.550. U.S. Department of Justice, 950 Pennsylvania Avenue, NW, Washington, DC.

Hicks, E. W. 2000. ``Protecting Construction Workers from Power Lines." *Job Safety and Health Quarterly*, No. 12 (Fall/Winter) 23-28.

Kisner, S and Casini, V. 1998. ``Epidemiology of Electrocution Fatalities." *A Summary of Surveillance Findings and Investigative Case Reports, National Institute for Occupational Safety and Health (NIOSH)* Pub. No. 98-131.

Mohan, S. 2000. ``A Device for Eliminating Electrocutions by Mobile Cranes." A Research Proposal Funded by *National Institute for Occupational Safety and Health (NIOSH)*, US Department of Health and Human Services, NIOSH Grant: 5 R01 OH03803-03, 2000-2003.

Mohan, S; Sivalenka, A; Coonly. E; and Wobschall, D. 2003. ``A Device for Eliminating Electrocutions by Mobile Cranes." *Working Paper, ViTech Systems Inc.,* Williamsville, NY.

Mohawk. 1990. "Overhead Electric Transmission Standards." *Niagara Mohawk*, 300 Erie Blvd West, Syracuse, NY.

Nguyen, D. H; Paques, J. J; Laliberte; and R. Bourbonniere. 1996. ``Numerical Analysis of Power Line Proximity Warning Device using Electric Field Measurement." *Journal of System Safety*, No. 32.

Occupational Safety and Health Administration. 2003. "Integrated Management Information System and Internal Database on Inspections." *Directorate of Information Technology*, 200 Constitution Ave N.W., # N3661, Washington DC.

Tamaki, Y and Yoshibumi, Y. 1996. "A 100 kV Transformer." *Technical Report 2, Vol. 77 , Misubishi Corporation,* Japan.

Department of Labor. 2000. "Table 1: Fatal Occupational Injuries by Event or Exposure, 1994-1999." *Census of Fatal Occupational Injuries*, Bureau of Labor Statistics, Washington, DC.

MODELLING OF LAYERED FLUID FLOW IN A CIRCULAR MICROCHANNEL

Manisah Aumeerally and Renate Sitte
School of Information Technology
Gold Coast Campus
Griffith University PMB50
Gold Coast Mail Centre
Queensland 9726, Australia
E-mail: m.aumeerally@griffith.edu.au

KEYWORDS
Microchannel, Micro-Electro-Mechanical system, flow model.

ABSTRACT

Microfluidic MEMS devices can be used in a wide range of applications. The design and manufacture of such devices use the same processing technique as in Integrated Circuits. MEMS are often structured into microchannels, microvalves, and micropumps. The flow characteristics of microchannels are important in the design of these devices. This paper describes the modeling of fluid flow in circular microchannels using an electrical network. It contributes to the physical component of our Virtual Reality-prototyping CAD tool for MEMS, with emphasis on fast calculations for VR representations.

With the underlying fluid flow model based on traditional continuum theory, the flow model is segmented into layered sections with different flow rate. This paper presents the models for the circular sections. The flow characteristics of each sections are modeled as resistors in an electrical circuit. Simulink (©Matlab) is used to simulate the behaviour of the model.

INTRODUCTION

Microfluidic devices such as microchannels, micronozzles, micropumps, micromixers, microfilters and microvalves involve the flow of fluid. These devices are often the components of an integrated microfluidic chip or "lab on a chip", forming part of a miniaturized chemical analysis system, a micro total analysis system or a micro dosing system as in Ikuta et al 2000, Koch et al 1999 and Nguyen et al 1998. These devices have dimensions in the range of 10 μm and are capable of delivering fluid in the nanolitres scale. Their application area includes the computer industry whereby smaller devices are essential to improve the resolution of inkjet printers. In the medical industry, they are used in laser ablation and drug delivery. Other application areas include pharmaceuticals, environmental maintenance, health care and analytical system for DNA sequencing. These microdevices have the advantage of producing high throughput, minute amount material requirement, low manufacturing, operating and maintenance cost, low power consumption, reduced waste, disposability and reduce risk of contamination, increased precision and accuracy.

With the rapid growth in the demand for such devices, there is a need to develop CAD tools that provide for quick and yet accurate design tool (Sitte 2003). Finite Element Modelling (FEM) software such as ANSYS Flotran™ provides detailed analysis of flow. This often requires hours of work involving the creation of the physical model, meshing, convergence monitoring and several iterations before satisfactory solution is obtained. The objective of this paper is to develop a quick first approximation model for the flow of fluid in a microchannel which will be used in an interactive Virtual Reality prototyping CAD tool (Aumeerally 2001). The underlying fluid model is based on the Navier-Stokes equations.

LAYERED FLOW MODEL

Fluid flow in MEMS is of low Reynolds number (Re) and at this miniscule dimension, viscous force dominates in relation to inertia while centrifugal forces are negligible. The flow is laminar and not turbulent. In long microchannels the flows are uniaxial, parallel to the wall. Important flow characteristics are viscosity, pressure loss and velocity distribution. In order to drive the fluid around the chip, a micropump is usually employed to provide the pressure driven flow.

For the VR CAD to display an animation of the flow for each layer of fluid in the microchannel, the different flow rate of each layer needs to be calculated. Laminar flow in a horizontal circular microchannel can be derived from the Navier-Stokes equation in cylindrical coordinates. This flow is given by (Fay 1994):

$$\left(\frac{d^2 V_z}{dr^2} + \frac{1}{r} \frac{dV_z}{dr} \right) = \frac{1}{\eta} \frac{dp}{dz} \qquad (1)$$

The solution to the above differential equation gives the velocity distribution:

$$V_z = \frac{a^2 - r^2}{4\eta}\left(-\frac{dp}{dz}\right) \qquad (2)$$

where a is the radius of the channel, η is the viscosity of the fluid and p is the hydrostatic pressure. The volumetric flow rate Q is found by integrating the velocity distribution across the cross-sectional area of the microchannel:

$$Q = \int_0^a V_z (2\pi r) dr \qquad (3)$$

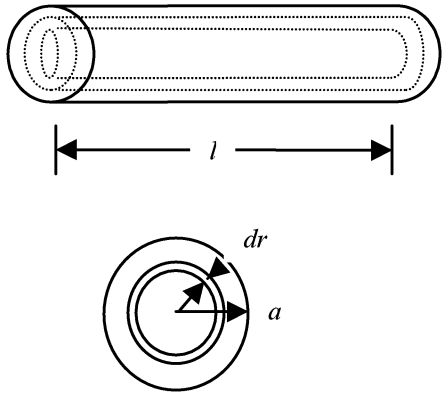


Figure 1 Sections in circular microchannel

The microchannel of length and radius a is divided into annular sections of width dr. The flow rate Q of each section is given by:

$$Q = \frac{\pi}{2\eta}\left(-\frac{dp}{dz}\right)\int_{r_1}^{r_2} r\left(a^2 - r^2\right)dr$$
$$= \frac{\pi}{2\eta}\left(-\frac{dp}{dz}\right)\left|\frac{r^2 a^2}{2} - \frac{r^4}{4}\right|_{r_1}^{r_2} \qquad (4)$$

where r_1 and r_2 are the inner and outer radii of the sections. For the lumped-element model of the mechanical effect of Poiseuille flow, using e→V convention, the resistance is given by (Senturia 2001):

$$R = \frac{across}{through} = \frac{\Delta P}{Q} \qquad (5)$$

The resistance for each sections can be determined by applying the above equation and is given by:

$$R_o = \frac{8\eta\ell}{\pi\left[2a^2\left(r_2^2 - r_1^2\right) - \left(r_2^4 - r_1^4\right)\right]} \qquad (6)$$

The flow rate for laminar flow in a circular pipe is given by the Hagen-Poiseuille equation (Zengerle and Richter 1994):

$$Q = \frac{\pi a^4}{8\eta L}\Delta P \qquad (7)$$

ELECTRICAL CIRCUIT MODEL

The concept of complex impedance have been used to model oscillating fluid flow in (Bardell and Forster 1998) and (Morris and Forster 2000), where the element impedances are:

$$Z_R = R, \quad Z_L = sL, \quad Z_c = \frac{1}{sC} \qquad (8)$$

In this model, the current represents the volume flow, the voltage is the pressure difference, the resistance is the viscous forces and inductance is the mass of the fluid. The resistance is given by:

$$R = \frac{\Delta P}{Q}, \; L = \frac{\rho l}{A} \qquad (9)$$

We consider the flow to be steady, so the voltage is a DC source and s = 0 for the above elements. Therefore:

$$Z_L = 0 \text{ and } Z_C = \infty \qquad (10)$$

We represent the layered flow for a liquid in a steady laminar flow by resistances in parallel:

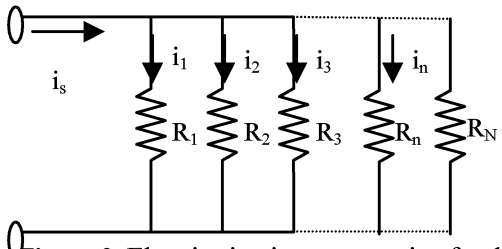


Figure 2 Electric circuit representation for the fluid sections in the microchannel

Each branch of the electric circuit in Figure 2 above represents the flow for each section in the microchannel. The current flow for each section is calculated by using current division rule:

$$i_n = \frac{1/R_n}{\sum_{n=1}^{N} 1/R_n} i_s \qquad (11)$$

This can be extended to the case of oscillating pressure flow in which case, AC analysis is used and the resistance *R* is replaced by impedance *Z*. The electric circuit will then include inductors in series with the resistors for each branch:

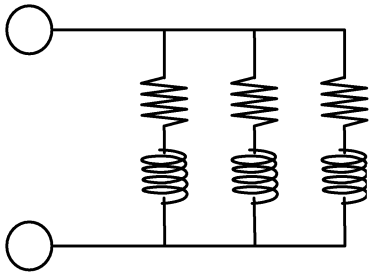


Figure 3 Resistance and inductor in series

APPLICATION EXPERIMENT RESULTS

As an example, we use a circular microchannel of length 4.8 cm and diameter 152 μm as in (Mala and Li 1999). This is divided into three annular layers of equal cross-sectional area. The resistances of each layer are calculated using equation (5). The electrical circuit model is implemented in Simulink (©Matlab) and the values of the current (i.e. flow rate) for each section will be used by the VR module. The length and radius of the microchannel and the pressure difference are inputs to the system. The flow rate for each layer is the output. Figure 4 shows the implementation of the current division rule in Simulink.

The flow rate for each layer obtained from the Simulink model and the values based on direct calculation of Hagen-Poiseuille equation, ie equations (2) and (6), are compared. This comparison is shown in Table 1. A finite element model of the microchannel is created using ANSYS. Meshing is done using Fluid element 141 and the axial velocities for the laminar flow of water in the channel is obtained. This is shown in Figure 5.

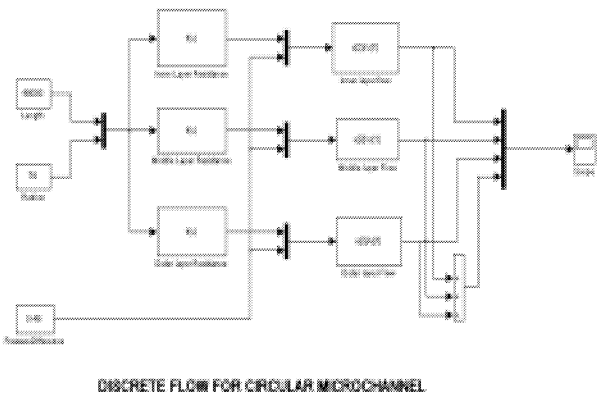


Figure 4 Simulink model for the current division for three sections in the microchannel

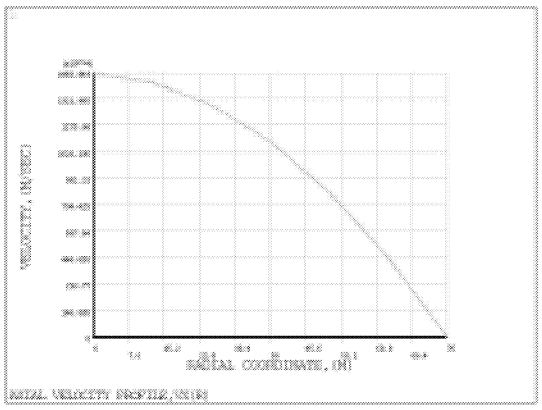


Figure 5 Contour plot for the axial velocity in a circular microchannel using ANSYS

VALIDATION AND DISCUSSION

From Table 1 and Table 2 it can be seen that the flow rates for the innermost layer is the fastest. Flow rate is dependent on the cross-sectional area and the average velocity of the fluid. (See equation (3)). In this analysis the areas of the layers are made equal so the flow rates are dependent on the velocity only. When compared with direct calculation using Hagen-Poiseuille equation, the electrical model gives an error ranging from 3.2% to 5.22%. When compared with finite element model (ANSYS), an error range of 3.05% to 6.58 % is obtained. It should be noted that ANSYS simulations may have an error of up to 10%. Our model is below this error.

Table 1 Comparison between electrical model and Hagen-Poiseuille equation of the layered flow rates for circular microchannel

Flow Rates	Circular microchannel		
	Electrical Model	Hagen-Poiseuille	% difference
Outermost layer	1.45e10	1.5299e10	5.22
Middle layer	4.35e10	4.494e10	3.20
Innermost layer	8.30e10	8.734e10	4.97

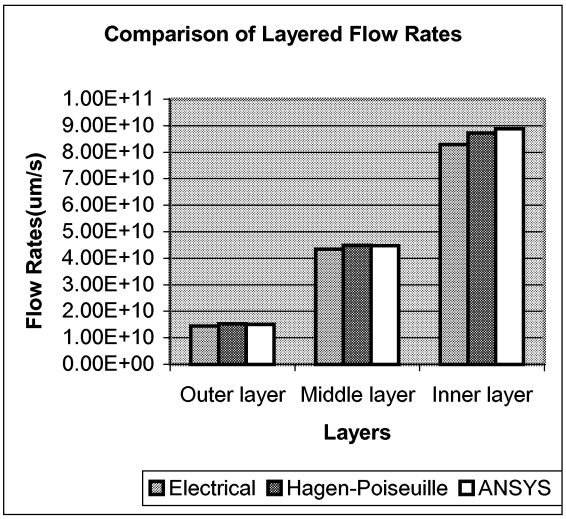


Figure 6 Comparison of layered flow rates using the electrical model, Hagen-Poiseuille equation and ANSYS

The advantage of using the electrical model is the time saved in calculations as well as in the preparation of the physical model and FEA meshing. However there are limitations. The electrical model is based on the analytical solution to the Navier-Stokes equation. As such, the solution for microchannel with regular shaped cross-sections can easily be obtained. Another limitation is that this analysis is carried out for a fully developed laminar flow, entrance and exit effects are not considered. The errors may be adversely affected if these effects are accounted for.

Table 2 Comparison between Simulink-electrical model and ANSYS of flow rates for circular microchannel

Flow Rates	Circular		
	Electrical Model	ANSYS	% difference
Outermost layer	1.45e10	1.5104e10	4.00
Middle layer	4.35e10	4.4867e10	3.05
Innermost layer	8.30e10	8.8845e10	6.58

CONCLUSION

In this paper we presented an electrical model for the fast calculation of the flow rates of fully developed laminar fluid flow in a microcahnnel. The model was implemented in Simulink and the results compared with direct calculation as well as with finite element model using ANSYS.

Compared with ANSYS, the electric network model in Simulink gives percentage errors of up to 6.6%. Compared with Hagen-Poiseuille equation, the percentage error is up to 5.22%. The electrical model provides a fast first approximation of the flow rate. There are no physical modeling or meshing to be done. This method is however restricted to devices such as microchannels of regular shapes in which the velocity profile can be obtained analytically from the Navier-Stokes equation. The analysis above is valid only for a fully developed laminar flow. The pressure losses due to entrance and exit effects have not been taken into consideration.

The electrical network concept of parallel resistances provides a suitable model for the flow rate of a liquid in laminar flow in a circular microchannel. Future work would involve incorporating the entrance and exit effects, curved channels as well as channels of other shapes, especially rectangular or square.

REFERENCES

Aumeerally, M. and R. Sitte. 2001." Design and Modelling of Microactuators in MEMS *Electronics and Structures for MEMS II, Neil Bergman, Editor, Proceedings of SPIE* 4591, 125-133

Bardell, R.L. and F.K.Forster. 1998. "Impedances for Design of Microfluidic Systems." *Proceedings of the Micro-TAS '98 Workshop* (Banff, Oct 13-16), Canada.

Fay, J.A. 1994. Introduction to Fluid Mechanics. MIT Press, USA.

Ikuta, K.; T. Hasegawa; T. Adachi; and S. Maruo. 2000. "Fluid Drive Chips Containing Multiple Pumps and Switching Valves for Biochemical IC Family." *Proceedings of IEEE Conference on MEMS* (Japan), IEEE, 739-744.

Koch, M; C. G. J. Schabmueller; A. G. R. Evan;s and A. Brunnschweiler. 1999. "Micromachined chemical reaction system." *Sensors and Actuators A: Physical,* Vol. 74 (Apr), Issues 1-3, 207-210.

Mala, Gh. M. and D. Li . 1999. "Flow Characteristics of Water in Microtubes." *International Journal of Heat and Fluid Flow*, vol. 20, 142-148.

Morris, C.J; and F.K.Forster. 2000. "The Correct Treatment of Harmonic Pressure-Flow Behavior in Microchannels." *Micro-Electro-Mechanical Systems* (MEMS vol 2), ASME 2000.

Nguyen, N.T.; S. Schubert; S. Richter; and W. Dötzel. 1998. "Hybrid-assembled Micro Dosing System Using Silicon-based Micropump/valve and Mass Flow Sensor." *Sensors and Actuators A: Physical*, Vol. 69, 85-91.

Senturia, S.D. 2001. "Microsystem Design." Kluwer Academic Publishers, USA.

Sitte R. 2003, "Introductory Physics Based Visual Simulation", *Proceedings of the C# and .NET Technologies 2003*, Plzen (Czech Republic), V.Skala (Ed), ISBN 80-903100-3-6, 63-69

Zengerle, R and M.Richter. 1994. "Simulation of Microfluid Systems." *Journal of Micromechanical Microengineering*, vol. 4, 192-204.

WEB BASED SIMULATION

PACKET DELAY MODELS IN PACKET-SWITCHED NETWORKS: PERFORMANCE ASSESSMENT THROUGH CAPACITY MEASUREMENTS

Leopoldo Angrisani
Dipartimento di Informatica e Sistemistica
Università degli Studi di Napoli "Federico II"
Via Claudio 21, 80125 Napoli,
Italy
E-mail: angrisan@unina.it

Salvatore D'Antonio
C.I.N.I.
Consorzio Interuniversitario Nazionale per l'Informatica
Via Diocleziano 328 ,80125 Napoli,
Italy
E-mail: salvatore.dantonio@napoli.consorzio-cini.it

Michele Vadursi
Dipartimento di Ingegneria Elettrica
Università degli Studi di Napoli "Federico II"
Via Claudio 21, 80125 Napoli,
Italy
E-mail: mvadursi@unina.it

Giorgio Ventre
Dipartimento di Informatica e Sistemistica
Università degli Studi di Napoli "Federico II"
Via Claudio 21, 80125 Napoli,
Italy
E-mail: giorgio@unina.it

KEYWORDS
Network Capacity Measurement, Link Bandwidth Measurement, Network Modelling, Packet-Switched Networks.

ABSTRACT
The paper presents a comparison of the performance of some software tools, mandated to bandwidth measurement in packet-switched networks. The considered tools are based on time measurements, according to specific packet delay models, already presented in the literature and briefly described in the paper.
An original measurement station has properly been set up by the authors with the aim of accurately performing the required time measurements, and then utilizing them as inputs for the tools. Final results are reported and compared.

INTRODUCTION

In order to meet bandwidth requirements of an increasing number of users, the Internet has significantly grown and been enhanced in the last years. As a consequence, the problem of bandwidth measurement, which had already arisen several years ago, is still topical for network monitoring.
With regard to a network path constituted of n link, named B_i, $i=1,...,n$, the link bandwidths, it is possible to define the *(end-to-end) capacity, C,* as

$$C = min(B_1, B_2, ..., B_n) \quad (1)$$

Among several software tools for link bandwidths measurement and end-to-end capacity, already presented in the literature, three of them have been selected: (i) *clink* (Downey 1999); (ii) *pchar* (Mah 1999) and (iii) *Pathrate* (Dovrolis et al. 2001). In particular, *clink* and *pchar* are based on the one-packet model, while *Pathrate* is based on the packet-train model.
The paper aims at comparing the performance of the aforementioned software tools in terms of measurement repeatability and compatibility. Measurements have been taken with regard to a particular network configuration, details of which are also given.
Being the considered tools based on time measurements, the authors have set up a proper measurement station, capable of carrying out the required time measurements by means of a digital counter. This choice allows the tools run with more accurate time estimates; thus, a more reliable comparison can be drawn.

THE ONE-PACKET MODEL

The One-Packet model predicts the delay experienced by a single packet across a path constituted of n links (Bellowin 1992; Jacobson 1997; Downey 1999; Mah 1999; Lai and Baker 2000).
Assumed the packet size is L, the time needed by the packet to reach its destination across n links is

$$T_n = \sum_{i=1}^{n} \frac{L}{B_i} + d_i + q_i \quad (2)$$

Equation (2) expresses the packet delay as the sum of the delays accumulated at each link. In particular, L/B_i is the transmission delay of the packet at link i; d_i is the latency of link i, i.e. the sum of propagation delay and other fixed per-packet delays due to the router; and q_i is the queuing delay experienced by the packet at link i.
It is worth highlighting some assumptions underlying the model: (i) the transmission delay is assumed to be linear with respect to the packet size; (ii) only store-and-forward routers are present on the path; (iii) links are single-channel.
With regard to the first assumption, it could happen that routers manage their buffers in such a way that the proportionality is not guaranteed. On the contrary, the second assumption is almost always verified on the Internet.
The last assumption is more critical. Some links, in fact, consist of multiple channels. In such cases, the application of the one-packet model could lead to wrong bandwidth estimation. As an example, let us suppose a link consists of M channels, each of which has a bandwidth of w bps (bits

per second), so that the overall link bandwidth is $M \cdot w$ bps. Since each packet is routed on a single channel, the bandwidth estimated by means of the one-packet model would be just w, rather than $M \cdot w$.
The presence of cross traffic can be responsible for the queuing delays q_i. If the network is probed several times, with packets of the same size, there is a high probability that the minimum value of delay is unaffected by cross traffic, i.e. $q_1 = q_2 = \ldots = q_n = 0$. Therefore, by considering the minimum value of delay over several observations for each packet size, the one-packet delay T_n results a linear function of the packet size, i.e.

$$T_n = a_n \cdot L + b_n \tag{3}$$

with

$$\begin{cases} a_n = \sum_{i=1}^{n} \frac{1}{B_i} \\ b_n = \sum_{i=1}^{n} d_i \end{cases} \tag{4}$$

Equation (3) is still valid if T_n is the Round-Trip Time (RTT) of a packet from the sender up to link n. In this case, the d_i terms also include the latencies on the way back and the transmission delay of the acknowledgement from link i, which is constant with respect to L.
With the aim of measuring the bandwidth of each link, Variable Packet Size (VPS) techniques probe the network with several packets for different values of L and select the minimum RTT for each considered value of L. A linear regression, performed over these values, provides the value of the slope a_n.
Figure 1 (Lai and Baker 2000) shows how queuing delays due to cross traffic can be filtered out from measurement results. Black dots represent the minimum delay for each probing packet size.

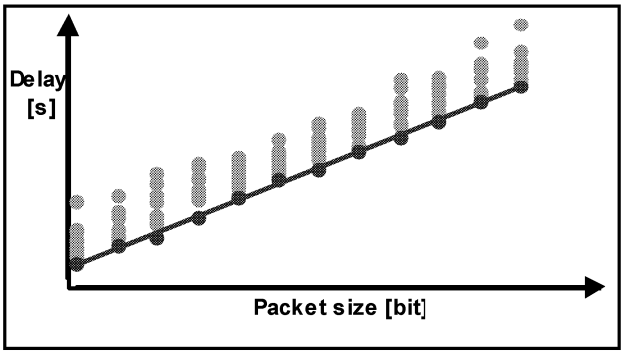


Figure 1: Filtering Cross Traffic Effects

An estimation of the link bandwidths $B_1,\ldots,B_n$ can be achieved recursively. For a one-link path, in fact, $B_1 = a_1^{-1}$. For a H-link path ($H>1$), it is easy to prove that

$$B_H = \frac{1}{a_H - a_{H-1}} \tag{5}$$

THE PACKET-PAIR MODEL

When the parameter of interest is the *end-to-end capacity* of the path (i.e. the *bottleneck* link bandwidth), rather than the bandwidth of each link, it is useful to refer to a different model, also presented in the literature: the *packet-pair* model (Bolot 1993; Carter and Crovella 1996; Paxson 1997; Dovrolis et al. 2001).
The model considers two packets of the same size L, travelling from the same source to the same destination, in the absence of cross traffic along the path.
Let us define the dispersion Δ_i at the generic link i as the time elapsed from the instant the last bit of the first packet is received at a certain path point to the instant the last bit of the second packet is received at the same point. Δ_i can be expressed as

$$\Delta_i = \max\left(\frac{L}{B_i}, \Delta_{i-1}\right) \tag{6}$$

Equation (6) predicts that if packets do not arrive at link i close enough to queue together ($L/B_i < \Delta_{i-1}$), their time spacing will not be modified. On the contrary, if they arrive at link i close enough to queue together, the new value of dispersion will be determined by the link bandwidth, and will result inferior to the previous.
Named Δ_0 the initial dispersion between the two packets, it is possible to express the dispersion at destination as

$$\Delta_n = max\left(\frac{L}{\min\limits_{i=1,\ldots,n} B_i}, \Delta_0\right) \tag{7}$$

Therefore, by measuring the dispersion between the two packets at destination, it is possible to estimate the end-to-end capacity, according to the definition given in (1).
It is worth stressing some assumptions underlying this model. As already seen for the one-packet model, the transmission delay is assumed to be linear with respect to the packet size, and routers are supposed to be store-and-forward. Moreover, bottleneck link router is supposed to adopt a FIFO (First-In-First-Out) routing policy. It is also necessary that the two packets are sent close enough to queue at bottleneck link.
Finally, a more limiting assumption is that the two packets queue together only at the bottleneck link. The presence of cross traffic could violate this assumption; the problem, however, could be solved by filtering out samples that have experienced additional queuing.

THE PACKET-TRAIN MODEL

As stated before, in the absence of cross traffic, end-to-end capacity can be estimated on the basis of the packet-pair dispersion measured at destination. On the contrary, the presence of cross traffic renders invalid Equation (6), i.e. the monotony of $\{\Delta_i\}$ is not guaranteed any more, and the distribution of the values of Δ_n measured at destination is not always unimodal. A direct extension of the packet-pair model, named packet-train model, takes specifically into account the effects of cross traffic (Dovrolis et al. 2001).

Considered a packet train of length N, the total dispersion of the train $\Delta(N)$ is defined as the dispersion between the first and the last packet in the train. Related authors have demonstrated that dispersion measurements for trains of increasing length converge to the estimation of the Asymptotic Dispersion Rate (ADR), which is lower than the capacity.
Suppose N packets of the same size L are sent back-to-back from a source to a destination. At destination, it is possible to estimate the bandwidth as

$$b(N) = \frac{(N-1)L}{\Delta(N)} \tag{8}$$

It is worth highlighting that in the absence of cross traffic, Equation (8) provides an estimation of the end-to-end capacity. On the contrary, the presence of cross traffic results in a multimodal distribution for $b(N)$.
Let us consider a one-hop network, with $B_0 > B_1$ and let r_1 be the average cross traffic incoming rate. The initial train dispersion is $\Delta_0 = (N-1)L/B_0$, while the dispersion at destination, due to the presence of cross traffic, is

$$\Delta_1 = \frac{(N-1)L + \overline{X}_1}{B_1} \tag{9}$$

where $\overline{X}_1 = \Delta_0 r_1$ is the average amount of cross traffic which arrives at the router during Δ_0.
The average bandwidth estimated at the receiver is, therefore,

$$R = \frac{(N-1)L}{\Delta_1} = \frac{B_1}{1 + u_1 \frac{B_1}{B_0}} < B_1 \tag{10}$$

where $u_1 = r_1/B_1$.
It is worth noting that Equations (9) depends on the average amount of cross traffic, but is independent of the distribution of cross traffic packets within the probing train, as well as their exact size. Upon N values's increasing, the distribution of R becomes unimodal and converges to the ADR. It is also evident that R reduces to the capacity when $u_1 = 0$, i.e. when there is no cross traffic.
Equation (10) can be generalized to the case of a H-hop path (Dovrolis et al. 2001).

NETWORK UNDER TEST AND MEASUREMENT STATION

In order to carry out the experiments, two couples of hosts have been utilized and a suitable network has been set up.
Figure 2 shows the network under test. Host A and Host B are connected to different Fast-Ethernet Local Area Networks (LANs), in particular to two different Cisco 3500 series switches. The switches provide an uplink to two Cisco 2600 series routers (Router 1 and Router 2), which are connected one each other.
The performance assessment is based on an extended experimental activity; about a hundred measurements have, in fact, been executed for each configuration. Experiments have been carried out with regard to two different bandwidth values for the connection between the routers: 100 Mbps and 2 Mbps.

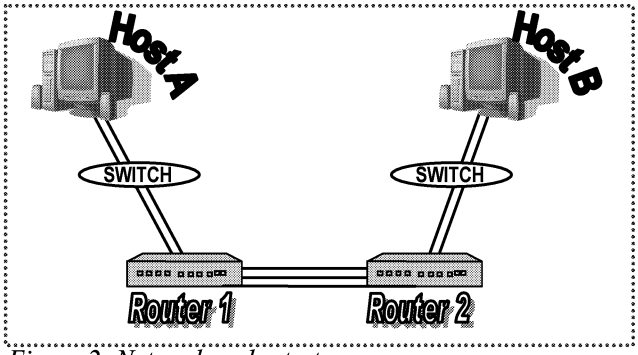
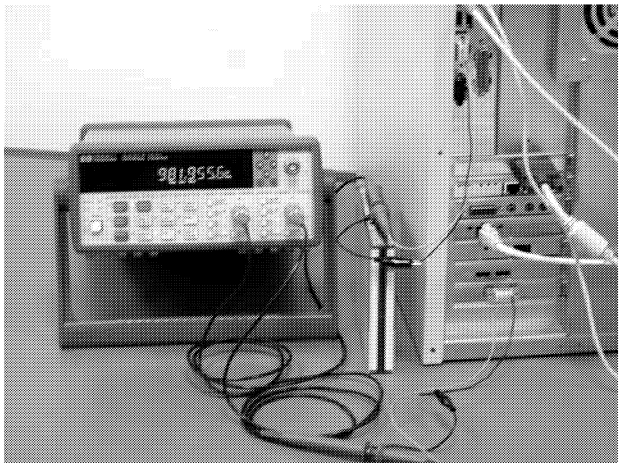


Figure 2: Network under test

The three tools have been utilized with the aim of measuring the capacity of the path from Host A to Host B. It is worth stressing that, while *Pathrate* measures the end-to-end capacity, *pchar* and *clink* measure the link bandwidths. As a consequence, with regard to *pchar* and *clink*, the capacity estimation has been achieved by measuring the RTT from Host A to Router 2 (i.e., because of the particular network topology, by setting the initial value of the Time-To-Live field equal to 2).
As stated before, considered models (and, consequently, considered software tools) derive an estimation of the bandwidth on the basis of measurements of time. Figure 3 shows the measurement station set up by the authors with the aim of providing software tools with more accurate time measurements.


Figure 3: Measurement station

The core of the station is a digital counter (HP 53131A). The two channels are connected to, respectively, pin 7 (RTS, Request to Send) and pin 4 (DTR, Data Terminal Ready) of the serial port of one of the hosts. In particular, the counter is connected to the host which performs the time measurements (i.e. the sender for *pchar* and *clink*, the receiver for *Pathrate*).
The digital counter provides the value of the time interval between the rising edges of two voltage pulses, which occur, respectively, on channel 1 and channel 2 of the counter. Adequate insertions have been made in the source

codes of the tools in order to generate the two voltage pulses at the right instants.
Time estimates attained by means of the described station are then given in input to the tools, and capacity values derived from these measurements are compared.

PERFORMANCE COMPARISON

Table 1 shows measurement results provided by the tools, for the two considered nominal path capacities (100 Mbps and 2 Mbps). Results are expressed in terms of mean value (μ) and experimental standard deviation (σ); the latter is given in percentage relative terms.

Table 1: Measurement results

	μ [Mbps]	σ [%]
	Nominal = 100 Mbps	
clink	96.50	0.14
pchar	95.40	0.17
Pathrate	96.80	0.15
	Nominal = 2 Mbps	
clink	1.98	0.29
Pchar	1.98	0.18
Pathrate	1.97	0.17

First of all, it is necessary to highlight that both *pchar* and *clink* provide a surprising result when the nominal capacity is 100 Mbps; specifically, estimated capacity is about half the nominal value. As proved by a recent work (Prasad et al. 2003), this is due to the presence of layer-2 store-and-forward switches on the network under test, which affects the validity of the model. Specifically, a layer-3 link with a store-and-forward Fast-Ethernet switch inside is equivalent to the series of two layer-2 segments of the same capacity, i.e. to a link of half the nominal capacity. According to these considerations, which definitely apply to our case, measurement results reported in Table 1 have been corrected.
The same problem does not arise in the 2 Mbps link, since no switch is present between the routers. Moreover, as Prasad et al. have proved, such an 'error' does not propagate on the following links.
Figures 4,5 allow a comparison of the results in terms of repeatability and compatibility with regard to, respectively, the 2 Mbps and the 100 Mbps link. The results of each tool are, in fact, expressed in terms of an interval centered at the mean value and whose width is six times the experimental standard deviation (a Gaussian distribution is assumed).
It is worth stressing that measurement results are compatible if the related intervals overlap. Repeatability, instead, is strictly connected to the relative experimental standard deviation; the smaller the experimental standard deviation is, the higher the degree of repeatability is.
From the analysis of the outcomes, the following considerations can be drawn:

- Measurement are compatible, since related intervals overlap for both the nominal capacity values.
- The experimental standard deviation is lower than 1% for the all considered tools.
- *pchar* and *clink* offer similar outcomes, although results provided by the latter presents a slightly greater standard deviation.
- Results given by VPS techniques (like *pchar*and *clink*) have to be corrected, in the presence of store-and-forward layer-2 switches. The one packet model, however, remains valid, if hops containing store-and-forward switches are modelled as a series of segments (Prasad et al. 2003).

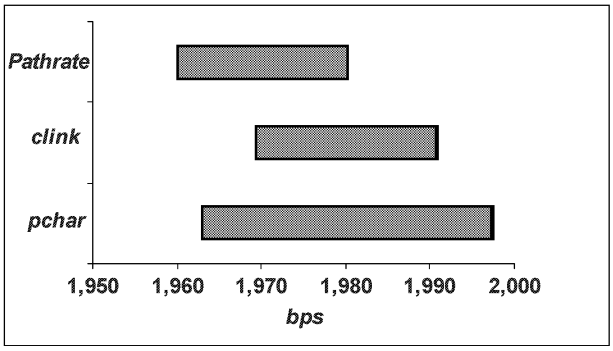


Figure 4: Measurement Comparison for a Nominal Capacity of 2 Mbps

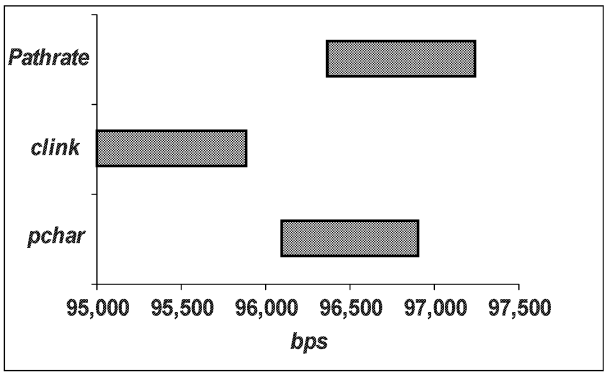


Figure 5: Measurement Comparison for a Nominal Capacity of 100 Mbps

For the sake of completeness, results provided by the tools, when time estimates are not attained by means of the measurement station, are reported in Table 2.

Table 2: Measurement results (II)

	μ [Mbps]	σ [%]
	Nominal = 100 Mbps	
clink	89.72	0.86
pchar	88.50	0.84
Pathrate	95.95	0.88
	Nominal = 2 Mbps	
clink	2.01	0.15
Pchar	1.98	0.11
Pathrate	1.97	0.12

By comparing results reported in the two Tables, it is possible to affirm that when time estimates are not provided by the digital counter, the tools exhibit worse

performance. In particular, the experimental activity shows that:

- Measurement results provided by the different tools are not compatible; intervals representing measurement results, in fact, do not overlap.
- Differences from nominal values are more marked. This is particularly true for the 100 Mbps link than for the 2 Mbps link.

CONCLUSIONS

The paper has compared the performance of different packet delay models for packet-switched networks. The performance assessment has been attained through some tools, mandated to bandwidth measurement, which implement the aforementioned models. A suitable measurement station, properly set up by the authors, has been presented, as well. The station provides accurate time estimates, which are then given as inputs to the tools.
To sum up some outcomes, provided by the experimental activity, it is possible to affirm that:

- Tools based on RTT measurements have proved to be unreliable in presence of layer-2 switches
- The adoption of a digital counter to carry out time measurements has clearly turned out to be a winning choice. Measurements, in this case, are evidently more compatible and repeatable; moreover, the difference between nominal and measured values tends to reduce.

REFERENCES

Bellovin, S.M. 1992. "A Best-Case Network Performance Model" http://www.research.att.com/~smb/papers/netmeas.ps.

Bolot, J.C. 1993. "End-to-End Packet Delay and Loss Behavior in the Internet". In *Proceedings of ACM SIGCOMM 1993.*

Carter, R.L. and M. E. Crovella. 1996. "Measuring Bottleneck Link Speed in Packet-Switched Networks". Technical Report BU-CS-96-006. Boston University

Paxson, V. 1997. "End-to-End Internet Packet Dynamics". In *Proceedings of ACM SIGCOMM 1997.*

Jacobson, V. 1997. "Pathchar: a Tool to Infer Characteristics of Internet Paths". ftp://ftp.ee.lbl.gov/pathchar/.

Downey, A.B. 1999. "Using pathchar to Estimate Internet Link Characteristics". In *Proceedings of ACM SIGCOMM 1999.*

Mah, B.A. 1999. "pchar: a Tool for Measuring Internet Path Characteristics". http://www.employees.org/~bmah/Software/pchar/.

Lai, K. and M. Baker. 2000. "Measuring Link Bandwidths Using a Deterministic Model of Packet Delay". In *Proceedings of ACM SIGCOMM 2000.*

Dovrolis, C.; P. Ramanathan; and D. Moore. 2001. "What do packet dispersion techniques measure?". In *Proceedings of IEEE Infocom 2001.* pp. 905-914.

Prasad, R.S.; C. Dovrolis; and B. Mah. 2003. "The effect of layer-2 store-and-forward devices on per-hop capacity estimation". In *Proceedings of IEEE Infocom 2003.*

AUTHORS' BIOGRAPHIES

LEOPOLDO ANGRISANI was born in Nocera Superiore, SA, Italy, on April 16, 1969. He received the M.S. degree (cum laude) in electronic engineering from the University of Salerno, and the Ph.D. degree in electrical engineering from the University of Napoli Federico II, in 1993 and 1997, respectively.
Since 2002 he has been Associate Professor at the Department of "Informatica e Sistemistica" of the University of Napoli Federico II. He is involved in research into new methods based on the wavelet and chirplet transforms for detecting, measuring, and classifying transient signals, new methods based on time-frequency transforms for testing RF equipment for mobile communications, new measurement procedures for communication networks test and measurement, and design, realization, and characterization of VXI instruments based on digital signal processors.

GIORGIO VENTRE is Associate Professor of Computer Networks in the Department of Computer Engineering and Systems of the University of Napoli Federico II. He owns a Laurea Degree in Electronic Engineering and a Ph.D. in Computer Engineering, both from University of Napoli Federico II. From 1989 to 1991 he worked at CPS, Center for Research on Parallel Computers of the Italian National Research Council (CNR), doing research in the area of system support for distributed memory computers. From 1991 to 1993 he was with the Tenet Group at the International Computer Science Institute and the University of California at Berkeley, working in the area of Real-Time Protocols for Multimedia Applications. Since 1993 he is back at the University of Napoli Federico II, where is co-leader of the COMICS team. COMICS stands for Computer for Interaction and Communications and is a research initiative in the areas of networking and multimedia communications. Recently Giorgio Ventre has been appointed as Director of ITEM, a research laboratory on multimedia application founded by CINI, the Italian University Consortium for Informatics. ITEM is located in Napoli and hosts a state of the art technological infrastructure in the area of telematics and multimedia systems. As leader of the networking research group at University of Napoli Federico II Giorgio Ventre has been main investigator for a number of national and international research projects, and in particular for BRAIN (EU RACE Programme), NICE, Renaissance, GESTALT (EU ACTS Programme), CADENUS, INTERMON, E-NET (EU IST Programme). Giorgio Ventre has co-authored more than 100 publications: he is member of the IEEE Computer Society and of the ACM. He has served in the Program Committees of international Conferences and Workshops and is reviewer for several international journals and conferences in the area of distributed systems and communication networks.

Website Migration Resource Scheduling of Adaptive Distributed Multimedia Web Servers

Mohammad Riaz Moghal and Mohammad Saleem Mian
Communication Systems Lab., Research Center, Department of Electrical Engineering,
University of Engineering and Technology,
Lahore-54890, Pakistan
E-mail: riazdat@yahoo.com; msaleem@iname.com

ABSTRACT

The majority of adaptive distributed multimedia web hosting (ADMWH) companies do not consider the use of expensive resource scheduling in their distributed multimedia web servers (DMWS). Website migration is a cost-effective and simple resource scheduling solution for ADMWH companies that host websites whose load is small in comparison to the capacity of a web server. The website migration is used to provide minimum the average latency and the variance in latency. The major costs, related to website migration are 1) interrupted service, 2) increased load on the web servers and the web server cluster's network, and 3) redirection. The main problem with website migration, as with any web server resource scheduling approach, is that resources are naturally multi-dimensional resources; scheduling approaches are hard to find. An innovative and fair algorithm for dynamic resource scheduling is presented. Due to random intervals of the newly accepted requests for websites hosting can cause some web servers to become highly loaded creating resource bottlenecks while others are lightly loaded or idle. Preferably, we would like to have all the web servers operating continuously for the website requests that eventually lead to the minimum execution time. The objective of resource scheduling is to enhance the performance of computations by automatically equalizing (minimizing bottleneck resource utilization (BRU)) the loads of servers during run-time. We know that dynamically minimizing the BRU will provide: efficient system utilization, more free resources for the new requests, which will eventually, improve the acceptance ratio of the new requests, and more availability of the system. Extensive simulation and their results show that website migration is a novel and fair approach for resource scheduling of web servers hosting small websites.

INTRODUCTION

The majority of the adaptive distributed multimedia web hosting companies specializes in offering low-priced web hosting services to owners of small websites. These "low-end" ADMWH companies do not use resource scheduling in their ADMWSs web servers. The idea is that they require simple and cost-effective web server clusters and the current resource scheduling approaches are expensive and complicated. We present a novel, cost-effective, fair and simple resource scheduler that can load share web servers that host websites whose loads are small in comparison to the capacity of a web server. The present approach is to find how to migrate websites among web servers using minimum BRU (Bokhari 1987; Dermler and Iqbal 1998; Moghal et al. 2003). Website migration resource scheduling is a kind of "resource scheduling problem," which was defined by researchers in (Steinmetz and et al. 1995; Wolf 1996; Bokhari 1987; Moghal et al. 2003). The chief benefits of website migration resource scheduling, in comparison to other approaches, are that it does not require expensive high-capacity resource- scheduling / load-balancer switches, or separation between front-end and back-end web servers.

We have investigated the potential of website migration resource scheduling using computer simulations. The rigorous simulation indicates that website migration resource scheduling can have significant positive results on the performance of web server clusters

The organization of the paper is as follows: second section discusses the present resource scheduling approaches and suggests a novel approach for resource scheduling in DMWS clusters. Third section formulates the problem and suggests solutions. Fourth section explains the experimental and simulation results. Fifth next section presents related works and finally sixth section discusses the conclusion of the paper.

RESOURCE SCHEDULING APPROACHES IN DMWS WEB SERVERS

Basic Concepts and Motivation

We discuss existing types of resource scheduling approaches in DMWS web servers. First is IP-Sprayer or One-IP approach (Linder and Shah 2002; Andresen and Yang 1997), which shares the load over front-end web servers using a resource scheduling / load balancing switch. The websites have multiple front-ends distributed over a series of front-end web servers. These front-ends handle the user website requests and can cache static data for efficiency. They communicate with the back-end servers on which all data is stored, so that the data remains synchronized among the duplicated front-end servers. The resource-balancing switch requires being of high capacity or it will become a bottleneck; it is therefore expensive.

The second, DNS resource scheduling approach (Linder and Shah 2002; Andresen and Yang 1997) is a bit variation from One-IP or IP-Sprayer approach and needs not a costly high capacity switch. The load is shared by the server cluster's DNS server, which alternates in its response among the IP-numbers of the front-end web servers that host the requested

website. The difficulty with this approach is that the IP-numbers are cached in higher-level DNS servers and at the client side over which the server cluster's DNS server has slight control. The IP-number can be cached for up to a couple of days.

The third approach uses resource scheduling of low-end ADMWH Clusters. Most of the ADMWH web servers hosting companies that are committed to small websites do not want to purchase and administer expensive high capacity switches or have separate back-end and front-end servers for their setups. This solution is to place a certain number of websites on each DMWS server and then manually move websites from DMWS servers that become overloaded. Website migration resource scheduling should enhance this process and make it automatic using adaptive and dynamic approach.

Adaptive Resource Scheduling of DMWS Web Servers

In an adaptive resource-scheduling problem of DMWS web servers, we consider, a number of web-servers equal m and number of websites equal n. The dilemma is that we are given sets L1,…, Lm, of m load vectors, that needs to be balanced, and that there is a significant cost associated with each move. A definition of the resource scheduling problem is: given m sets L1,…, Lm, of positive and rational d-dimensional vectors, moves k vectors, so that the maximum sum of each set L1,…, Lm in any dimension is minimal. The resource-scheduling problem is also NP-complete (Bokhari 1987) and difficult to solve. Approximate load-balancing/resource-scheduling algorithms were proposed in (Dormer and Iqbal 1998; Moghal et al. 2003). Agawam and et al proposed an algorithm of complexity O (m * log (n)), where m is the number of web servers and n is the number of websites. This algorithm assumes each website as a one-dimensional load index and a single website is being allocated / migrated on a single server.

We present more a flexible model and algorithm for this kind of problem. Here we consider multiple websites' requests with multiple resource-constraints could be allocated or migrated onto a lightly or idle server. This is a MMKP problem which is an NP-complete problem and could be solved by our presented heuristic algorithm REMWS (REduced bru Multiple Web Server) (Moghal et al. 2003).

A website experiences by downtime when being moved and a minor increase in latency from redirection until the cached DNS entries for the website have been updated with the IP number for the new web server. If the resource scheduler would move one particular website often the owner of that website could become annoyed. Algorithms that treat websites fairly are therefore preferred. Fairness made us discard the idea of taking into assumption that a website that is, at the time of the move, already experiencing from redirection will not experience as much, in terms of redirection, as a website for which all cached DNS entries are consistent. If we were to choose to move websites that were most recently moved, then some websites could come to suffer from being under constant redirection.
The chief objective of web server resource scheduling is to decrease the average latency and the variance in latency. A website request that is rejected because of overload can be seen as having infinite or some maximum latency. There are substantial costs involved with relocating a website. The most important costs are: 1) interrupted service, 2) increased load on the web servers and the web server cluster's network, and 3) redirection. While a website is moved it must be put out of service. This would be unwanted for a commercial website with millions of customers, but "low-end" web hosting companies are only concerned with small websites whose owners do not need 100% uptime. As far as a website is up almost all of the time its owner will be satisfied. The cost of downtime can be defined in terms of the number of hits that were rejected while the website was out of service.

The increased load on CPU and disk from zipping and unzipping the virtual machine of the website is heavy and lasts for several seconds. The latency for the requests that are controlled during the time of the removal or installation for almost all cases will considerably larger than if the website have not been moved.

When a website is moved the DNS must be modified. Until the caches of the higher-level DNS servers have been modified, that can take somewhere from an hour to a few days, a redirection must be left at the web server on which the website was hosted before the move. Observe that this redirection must be modified if the same website is moved once again, so that the user is not redirected multiple times. The redirection can be removed after a week. This cost can be defined as the latency caused by the redirection of requests. After a website has been moved from a heavily loaded web server to a less heavily loaded web server, there will, if the measurements of the resource scheduler are correct, be a reduction in average latency and a variance in latency. Measuring the costs and benefits is very hard. There are many variables and many of them are hard or impossible to measure. If the costs are underestimated the resource scheduler could have a negative effect on the cluster of web servers that it services. A pessimistic approach should therefore be taken when calculating the costs.

We propose a novel approach for maximizing the systems service under the given system resource (CPU, Main Memory, I/O bandwidth, and Network) constraints and a model for ADMWH model for our approach (Fig 1). The random intervals of the newly accepted requests for websites hosting can cause some servers to become highly loaded creating bottlenecks while others are lightly loaded or idle. Preferably, we would like to have all the web servers operating continuously for the website requests that eventually lead to the minimum execution time. The objective of resource scheduling is to improve the performance of ADMWH by automatically equalizing (minimizing bottleneck resource utilization (BRU)) the loads of servers during run-

time (Bokhari 1987; Moghal et al. 2003). The QoS of each website request in this approach is adapted to dynamic changes of available resources based on distributed loads and users choices.
As dynamically minimizing the BRU will provide:
1) efficient system utilization, 2) more free resources for the new website requests, which will eventually improve the acceptance ratio of the new website requests, and 3) more availability of the system.

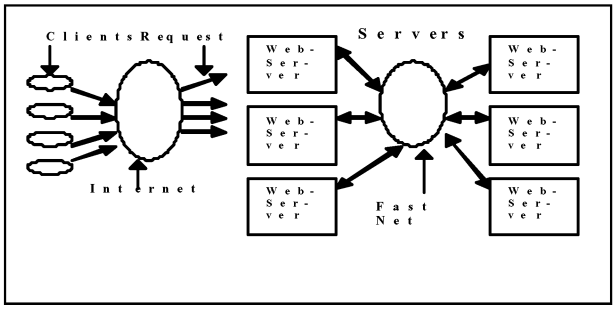


Figure 1 Adaptive Distributed Multimedia Web Servers Hosting (ADMWH)

PROBLEM FORMULATION AND SOLUTION

We present more flexible model and algorithm for this kind of problem. Here we consider overloading of server(s) due to newly accepted arrival of multiple websites' request(s) with multiple resource-constraints. This is a MMKP, which is an NP-complete problem, and is solved by our heuristic algorithm REMWS.

In an adaptive website migration resource scheduling problem of an ADMWH web servers, there are: m number of web servers, n number of websites on one server, and k is the number of servers in ADMWH servers' cluster. This is a multiple chain (of websites)-on-multiple-chain (of web-servers) or chain-of-tasks over chain-of-web-servers like problem (Bokhari 1987; Moghal et al. 2003). The problem can be solved by partitioning / chain-cut approach as shown in Fig 2. In Fig 2, we assume that websites on a server forms a chain. And we further assume when website(s) are required to be moved from overloaded server(s) to under-loaded or idle server(s). This decision is like finding a path or chain-cut in the problem. We solve this resource scheduling problem by our heuristic algorithm ***REMWS***.

A resource scheduler in the system has the following features: 1) the users have to state their QoS requirements and choices in an adaptable way, 2) the system has to know the existing status of its resources, 3) the system has to have flexibility to accept or reject the website request, 4) the system must have the flexibility of drop off a website request, and 5) the system must be flexible in providing QoS adaptation, priority-based resource allocation and integrated resource management so as to make best possible use of the available system resources (Lee et al. 1999; Khan 1998; Yamazaki and Matsuda). The multiple concurrent requests are dynamically adapted to the available resources and user choices in an ADMWH. Such models and algorithms for resource scheduling for ADMWH have been discussed in (Moghal et al. 2003). It presents a unified and feasible way to solve the admission problem for newly ADMWH website request, and the dynamic QoS adaptation and integrated resource allocation problems for existing requests (Moghal et al. 2003; Steinmetz et al. 1995).

Assume a user creates a website request i which has an operating quality q_i in an ADMWH environment. We map from an operating quality to the required resources for a

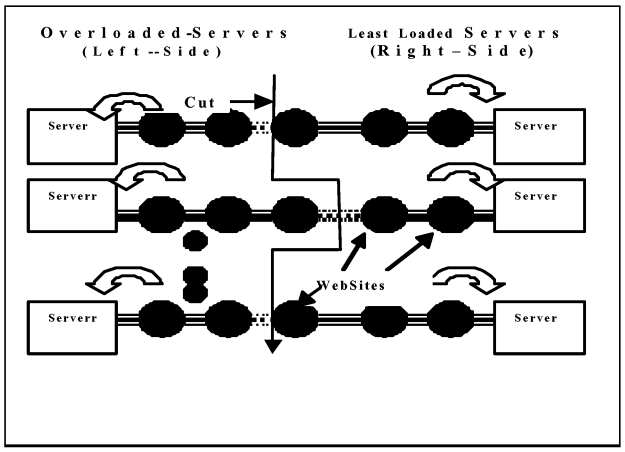


Figure 2: Partitioning / Chain-Cut of Websites Loads onto Web Servers Machines

particular quality request. The resources required by the website request are represented by $R(q_i)$ where $R(q)$ denotes the quality to resource mapping (***M:*** Q → R) function.

An adaptive resource scheduler must have certain targets to determine the adaptation of the system in a particular circumstance. The goal of a resource scheduler is to maximize the system serviceability. The system serviceability is equal to sum of all the individual requests utilizations. The service function-S of a system maps a website-request's operating quality q_i to website-request service $s(q_i)$. We suppose that system service objective is to maximize the system serviceability function S known as:

$$S = \sum s_i(q_i)$$

Here the system serviceability or utilization is the total profit earned by the system. We know the system state must consider the system resource constraints at every time. The sum of the quantities of the resource scheduled to all the requests must not exceed the total available quantity of the resource. If R_T is the available system resources then the resource constraints are defined as:

$$\sum R(q_i) \leq R_T$$

The main problem in an adaptive resource scheduler of ADMWH web servers is to work out the operating quality q_i of each website request i which maximizes the system service S by minimizing the BRU under given system resource constraints. The minimization of the BRU by using different approaches is discussed in (Bokhari 1987; Moghal et al.

2003). This minimizing BRU approach improves both: 1) the website-request acceptance ratio and 2) response of the system.

For simplicity, consider websites' requests, and we want to schedule the resources for load balancing in an ADMWH onto web-servers. We take four different cases for our solution to the problem:
1) If we find a single web-server which is least–loaded and victim for website(s) migration. This is similar to single-server-client/single-host-satellite distributed architecture problem and can be solved by SQ algorithm (Bokhari 1987; Dermler and Iqbal 1998; Moghal et al. 2003). 2) If we find two web-servers which are least-loaded and victim for website(s) migration. This is similar to double-server-client/double-host-satellite distributed architecture problem and is solved by our RED algorithm (Moghal et al. 2003). 3) If we find three and web-servers which are least-loaded and victim for website(s) migration. This is three/multiple-server-client distributed architecture problem and is solved by our present REMWS algorithm (Moghal et al. 2003). 4) If we find four web-servers which are least-loaded and victim for website(s) migration. This is multiple-server-client problem distributed architecture and solved by our present REMWS algorithm (Moghal et al. 2003).

The Service Model can be used for website request admission, dynamic quality adaptation, integrated resource scheduling, and priority-based resource allocation in distributed real-time ADMWH web servers systems. The terms website request acceptance, QoS adaptation, integrated resource management and priority based resource allocations are discussed (Moghal et al. 2003).

SIMULATION AND RESULTS

REMWS: Our proposed algorithm REMWS calculates initial partitions / cuts using an efficient greedy approach. It gradually improves the cuts and eliminates bad alternatives. This yields the lowest BRU. The lowest BRU corresponds to a high acceptance ratio to new website request with more responsiveness of the system. Then REMWS-algorithm considers several iterations. Both the best and the worst cuts in terms of BRU are derived for iteration. The worst case is disabled and is not further considered during next iteration. One cut is disabled during iteration.

Rigorous simulations are performed for all four cases discussed in previous section. We consider random capacities of web servers and random load of each website. We have considered average load (CPU, Memory, and Network Bandwidth) for each website. We have implemented and evaluated our proposed algorithm for various problem sizes. Results are shown and compared on the basis of Average Relative BRU for the different problem sizes and for 100 iterations. For these problems we have varied the number of victim or least loaded server for website(s) migration as 1, 2, 3, and 4 as discussed in previous section. Our algorithm finds the best solutions in terms of minimum BRU compared

REMWS Algorithm (For Reference See Fig 2.):

```
Input: c = # of web servers which are overloaded;
       = # of chains (left-side of chains are attached with the overloaded web servers
     m. = # of cuts in each chain; each cut with a 2k-dimensional load-vectors
      Left-side-of-chain-cut for overloaded-web-servers and Right-side-of-chain-cut for
leastloaded-web-Servers
(in our cases leastloaded-web-servers = 1, 2, 3 and 4)
                k = # of constraints
              L = normalization value (Between 0 to 1)
Overloaded-web-serverId =Cp, where C=# of web-server which are overloaded; 1≤p≤c
 n leastloaded-web-Servers = S1-Server1; S2-Server2,... ,Sn-Server n
Output: Set of Cuts { }
                        Bottleneck resource usage BRU
                        Bottleneck resource id BR = Best Cut
Step (A): The overloaded-web-server and leastloaded-web-servers S1 or S2 or ... Sn, load
initialization.
          for all Cp, set used load-vector to 0
                for S1 or S2,...Sn set used load-vector to 0
Step (B): Calculate initial cuts for chains
          for (i=1; i <= c; i++){
                        for (j=1; j <= m; j++){
          vector-add load Vector of chain i at cut j to used-
          -load of Cp and S1 or S2, ... Sn
            get the current BRU
            if current BRU lower than best BRU so far
                  store j as best cut so far and update best BRU
                  remove load of chain i from Cp and S1 or S2 or...Sn
            }
vector-add load of chain i at the best cut found to used load of Cp and S1 or S2 ... Sn
    }
 Step (C): improve iteratively the selected cuts
          for( iteration=1; iteration=m-1; iteration ++){
                        for (i=1; i <= c; i++) {
          remove load of chain i from Cp and S1orS2 or... Sn as implied by best cut so far
          find best cut & worst cuts from chain i as in
          Step (A)
                        }
                        mark worst cut as disabled
      vector-load of chain i. at the best cut found to used load of C and S1 or S2 ...Sn.
    }
Step (D): compare result
For (i=1; i≤c; i.++)
    C = best cut for Best-Load-Balance after last iteration of Step (C)
   BRU = bottleneck resource usage after last iteration of step(C)
```

to SQ and RED for different problem sizes. This shows that our REMWS algorithms are more dynamic to handle the dynamic situations of the website requests for DMWS web servers. The simulation results also show that in terms of minimum relative BRU, our algorithm with three-web-servers outperforms with 9%-11.31% than RED and 30.30-31.29% than SQ algorithms with different problem sizes. Our algorithm with four-web-servers outperforms with 27-28.8%

than RED and 44.93-44.45% than SQ algorithms with different problem sizes. Further, simulation results are presented in Fig 4, 5 and 6 that also support performance of our algorithm. We have performed the simulations on Pentium-II machine, 128-MB RAM using C++. The results for minimum BRU of REMWS algorithm for multiple-web-server architecture are very promising.

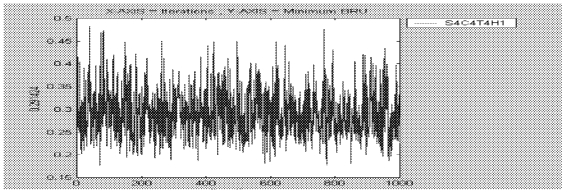


Figure 3: BRU-Simulation for Least Loaded Web Server =1 and Over Loaded Web Servers =4

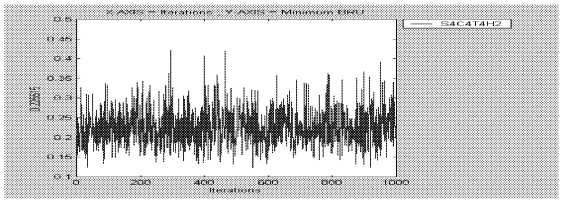


Figure 4: BRU-Simulation for Least Loaded Web Server = 2 and Overloaded Web Server =4

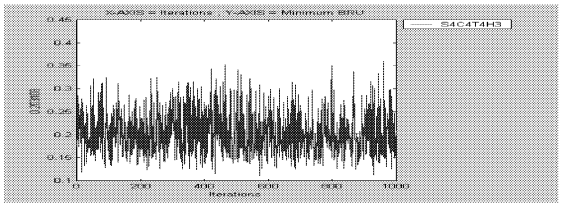


Figure 5: BRU-Simulation for Least Loaded Web Server =3 and Overloaded =4

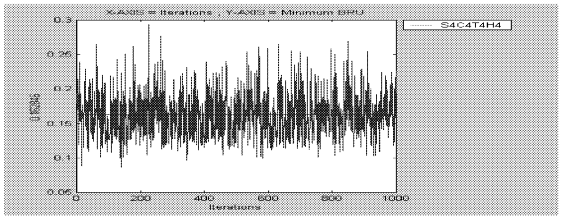


Figure 6: BRU-Simulation Least Loaded Web Server = 4 and Overloaded Web Server = 4

RELATED WORK

In an adaptive resource-scheduling problem of web servers, there are m number of web servers and n number of websites. The problem is that we are given sets A1,..., Am, of m load vectors, that need to be balanced, and that there is a significant cost associated with each move. A definition of the resource scheduling problem is: given m sets A1,…,Am, of positive and rational d-dimensional vectors, move a k vectors, so that the maximum sum of each set A1,…,Am in any dimension is minimal. The resource-scheduling problem is an NP-complete and difficult to solve. Approximate resource scheduling algorithms were proposed in (Bokhari 1987; Dermler and Iqbal 1998; Moghal et al. 2003). The greedy algorithm proposed by Aggarwal et al. takes into consideration the major cost associated with each move by maximizing the number of moves to k (Linder and Shah 2002). Where, k could be made proportional to the measured imbalance of the system, which would reduce the resource scheduling cost when there is little to gain. This algorithm has complexity O (m * log (n)), where m is the number of web servers and n is the number of websites. The algorithm considers only one-dimensional load and considers one website migration and n total number of websites. The model and algorithm presented by the authors are not flexible. We consider adaptive and flexible model and algorithm for solution of the problem.

CONCLUSION

The current web server resource scheduling approaches are very costly. Website migration is a cost-effective and simple resource scheduling solution for ADMWH companies those host web sites whose load is small in comparison the power or capacity of a web server. Our rigorous simulation results show that website migration is a novel approach for resource scheduling of web servers hosting small websites.

Website requests enter the web servers at a greater rate than the web server can complete it, causing a number of connections at the server to build up. This implies large delays for clients accessing the server. We present a dynamic task allocation model for scheduling requests, optimizing the use of client resources as well as the distributed web servers. This model provides a scheduling mechanism for distributed servers incorporating dynamic load distribution and distributed data access. In this paper we present a novel algorithm, REMWS for multiple-web-server-client architecture for website migration resource scheduling. The main objectives of our algorithm are (1) to make the load balance as good as possible, (2) to respond to a new web request from web applications as soon as possible, 3) improves request-acceptance-ratio, and 4) improve overall systems-utilization/system-benefit, The simulation for REMWS algorithm has been performed by varying both the number of servers and the websites loads on each web server. The simulation results show that in terms of minimum relative BRU, our algorithm outperforms than SQ and RED algorithms.

REFERENCES

Steinmetz, R. and Nahrstedt, K. 1995. "Multimedia: Computing, Communication, & Applications", Prentice Hall, Innovative Technology Series.

Linder, P. and Shah, A. 2002. "Website Migration Load Balancing of Web Servers".

Wolf, L. 1996. "Resource Management for Distributed Multimedia Systems", Kluwer Acad. Publication Boston.

Dermler, G. and Iqbal, M. A. 1998. "Task Assignment in Distributed Multimedia Systems", Technical Report, University of Stuttgart/IPVR, Germany.

Bokhari, S.H. 1987. "Assignment Problems in Parallel and Distributed Computing", Kluwer, Boston.

Lee, C. et al. 1999. "A Scalable Solution to the Multi-Resource QoS Problem", CMU-CS-99-144.

Khan, S. 1998. "Quality Adaptation in a Multisession Multimedia System: Model, Algorithms and Architecture", Ph.D. Dissertation, Department of ECE, University of Victoria

Yamazaki, T. and Matsuda, J. "On QoS Mapping in Adaptive QoS Management for Distributed Multimedia Applications", ATR Adaptive Communications Research Laboratories, 2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0288 Japan.

Moghal, M. R. Mian, M. S. and Satti, J. A. 2003. "Adaptive QoS-Based Load-Sharing in Multisession-Video-Conferencing Distributed Multimedia System", DMS 2003, The 9th International Conference on Distributed Multimedia Systems, FIU, Miami, Florida, USA, (Sep)

Andresen, D. Yang, T. 1997. "Multiprocessor scheduling with client resources to improve the response time of WWW applications". In *Proc. of the 11th ACM/SIGARCH Conf. on Supercomputing(ICS'97)*, Vienna, Austria, (July)

THE EFFECT OF TRAFFIC ENGINEERING IN PLANNING AND CAPACITY SCALING ON INTERNET PLATFORMS

Gerhard Haßlinger* and Stefan Schnitter**

*) Darmstadt Univ. of Tech., Dept. of Computer Science, Wilhelminenstr. 7, D-64383 Darmstadt, Germany

**) Deutsche Telekom, T-Systems, Am Kavalleriesand 3, D-64307 Darmstadt, Germany

E-mail: hasslinger@iti.informatik.tu-darmstadt.de, stefan.schnitter@t-systems.com

KEYWORDS

Network planning, traffic engineering, optimized load balancing, resource utilization

ABSTRACT

The routing of data in IP (Internet protocol) networks usually follows the shortest path principle based on link weights. The introduction of MPLS (Multiprotocol label switching) raised new opportunities for support of network planning, including measurement of the traffic matrix and the establishment of explicit data paths (LSP: label switched path) form source to destination. This allows to optimise the LSP design in order to balance the traffic load on the network links.

Making use of previously developed optimisation algorithms, we study the gain of traffic engineering during a continuous upgrade process of link capacities according to the increase of Internet traffic. We obtain an upper bound of the possible gain based on a modelling approach and compare the results to simulation of a realistic scenario.

INTRODUCTION

In this paper we investigate the benefit of traffic engineering in terms of reducing the maximum load on network links and thus the link capacities to be installed in a telecommunication network. In particular, a network topology is given by a graph $G = (V, E)$ with a set $V = (v_1, \ldots, v_N)$ of nodes and $E \subseteq V \times V$ of unidirectional links with capacities c_{jk} for each link $e_{jk} \in E$ and a predefined traffic demand matrix $T = (t_{jk})$ of flows to be delivered from node v_j to v_k. Then we determine maximal real valued factors f_{SFP} and f_{TE} such that the demand $f_{SFP}\,T$ ($f_{TE}\,T$) can be delivered through the network under the shortest path first (SPF) principle or with optimised path selection supported by traffic engineering (TE), respectively. For both cases, SPF and TE, each demand t_{jk} is allocated on one or several paths through the network from source to destination. In MPLS (Multiprotocol label switching) networks [14] the complete set of provided paths is denoted as an LSP (Label switched path) design. Then the gain g_{TE} obtainable by an optimised LSP design is defined as the ratio $g_{TE} = f_{TE} / f_{SPF}$.

When applying optimisation algorithms developed in [9][15] to scenarios which are typically observed in service provider networks, we experienced TE gains in the range of about $1.25 < g_{TE} < 2.5$. The gain obviously is related to network planning and the process of extending link capacities to cope with steeply increasing traffic demands on Internet platforms and depends on the extend to which bandwidth can be exploited already without optimised load distribution. We refer to [5][6][7][14] for general aspects of MPLS TE.

If we assume stable traffic flow conditions in normal operation for a network dimensioning focused on daily or weekly peaks, i.e. if the fluctuations of the peak rates from day to day (week to week) are relatively small for each traffic demand except for a steady and uniform increase, then major gaps of underutilized bandwidth are often encountered on a link after its capacity has been upgraded. Upgrades to double or even four times the capacity are usually provided at points in time when the link utilization exceeds a threshold, which is set to guarantee sufficient QoS (quality of service) and reliability for applications over the communication network. Then the link utilization is dropping down to half or a forth of the threshold due to the SPF principle without further optimisation. Traffic engineering on the other hand can fill up this gap by reoptimization of the LSP design after each upgrade, such that newly installed bandwidth is immediately used in order to reduce the load on other links [16].

In the next section we derive an upper bound of the possible improvement of link utilization by TE due to this effect as a main contribution to TE business case studies. Then TE algorithms are briefly summarized and we compare the resource efficiency of SPF and TE in examples, which indicate how much of the possible gain can usually be realized.

MODELLING TRAFFIC DRIVEN EXPANSION IN IP NETWORK PLATFORMS

Utilization for link upgrades with SPF

In recent years the traffic volume transported over the Internet is increasing at a high rate. This situation demands for a steady adaptation of the transport capacities. We start from a predefined network topology with nodes, links and capacities on each link and the traffic demand matrix as described in the introduction.

Some simplifying assumptions are made for modelling the planning and link extension process on Internet platforms

- All traffic demands are exponentially increasing over time. Starting from $T(0) = (t_{jk}(0))$ the demand from source v_j to destination v_k at time x is $t_{jk}(x)) = t_{jk}(0)) \exp(\omega x)$ where ω is a unique common increase rate.
- A link is upgraded by a factor $F_{Upgrade} > 1$ each time when its utilization $u_{jk}(x)$ reaches a predefined threshold $T_{Upgrade} < 1$. The utilization $u_{jk}(x)$ and the installed capacity $c_{jk}(x)$ are also variable over time.

Under this standard rule for link extensions, the utilization periodically increases from $T_{Upgrade} / F_{Upgrade}$ to $T_{Upgrade}$ and again drops down immediately after the next upgrade as shown in figure 1. The example in figure 1 assumes a 2.5-fold increase per year over a 3 year or 156 week period. The standard link extension achieves a mean utilization of

$$\overline{u}_{jk} = \frac{T_{\text{Upgrade}}}{F_{\text{Upgrade}}\, I_{\text{Upgrade}}} \int_0^{I_{\text{Upgrade}}} \exp(\omega x)\, dx \qquad (1)$$
$$= T_{\text{Upgrade}} \frac{1-1/F_{\text{Upgrade}}}{\ln(F_{\text{Upgrade}})}; \quad I_{\text{Upgrade}} = \frac{\ln(F_{\text{Upgrade}})}{\omega}$$

during a period of length I_{Upgrade} between successive upgrades and thus asymptotically over long time scales. Note, that the result is independent of the parameter ω expressing the intensity of traffic increase. Link capacity in IP backbone networks is provided over optical transmission lines in the STM (Synchronous transmission module) framework at steps of a 4-fold capacity increase (STM-1 ≈ 622 Mbit/s; STM-4 ≈ 2.5 Gbit/s; STM-16 ≈ 10 Gbit/s; STM-64 ≈ 40Gbit/s etc.). Upgrade steps to twice the capacity are also usual with bundles of two ports of the same capacity in intermediate steps.

The utilization gaps after each upgrade are still visible in the mean utilization computed from (1) and depicted in figure 2 as a function of F_{Upgrade}. This shows that the mean utilization is reduced by about 28% for $F_{\text{Upgrade}} = 2$ and to almost half of the threshold for $F_{\text{Upgrade}} = 4$, which on the other hand provides a potential to be exploited by traffic engineering.

If capacity extension can be carried out within a set of different factors $\{F_1, \ldots, F_M\}$ and if the probabilities $q_m = \Pr\{F_{\text{Upgrade}} = F_m\}$ are predictable, then the mean utilization is obtained on a link or network wide by the following generalization of equation (1), where each case for F_m is weighted by the length $\ln(F_m/\omega)$ of its upgrade period

$$\overline{u} = T_{\text{Upgrade}} \sum_{\text{m}} q_m (1-1/F_m) \Big/ \sum_m q_m \ln(F_m) \qquad (2)$$

TE support to keep the utilization close to the threshold

When traffic engineering is applied then link upgrades are not only locally relevant for the concerned link. After a reoptimization step, which should be done shortly after each upgrade or change in the topology, an almost full utilization of the new installed capacity can be achieved while the load on other links in the neighbourhood is reduced accordingly. We refer to section 3 for a closer look on the algorithms and properties applying to load balancing. Considering the utilization on a link $e_{jk} \in E$, it is decisive how many parallel and link disjoined paths can be established from node v_j to v_k through the network, such that traffic load on e_{jk} may be shifted within the set of such paths. Let p_{jk} denote the total number of parallel and link disjoint paths from v_j to v_k including e_{jk}. Then the mean utilization during an upgrade interval of length $I_{\text{Upgrade}} = \ln(F_{\text{Upgrade}}/\omega)$ for the link e_{jk} is maximized,

- when the time points of upgrades which extend the capacity on one path in the set are equidistantly interspaced by $I_{\text{Upgrade}}/p_{jk}$ and
- when after each upgrade the load on the link e_{jk} is reduced by the same factor

$$F_{\text{Upgrade, TE}} = \sqrt[p_{jk}]{F_{\text{Upgrade}}}.$$

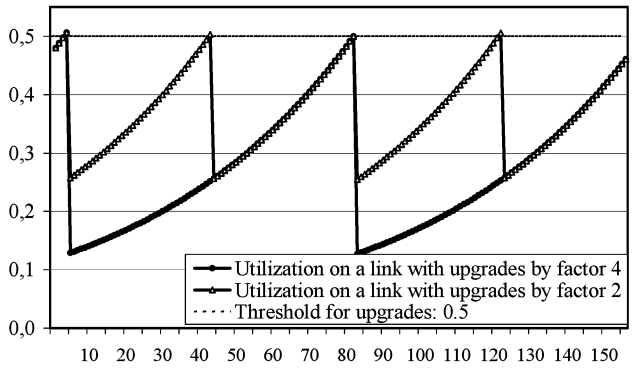


Figure 1: Utilization gaps after link upgrades

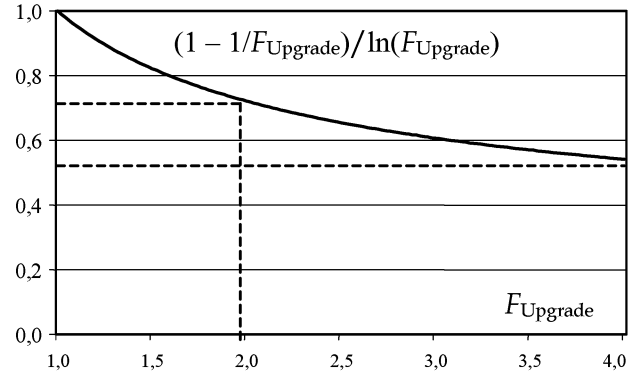


Figure 2: Mean reduction of the utilization for link upgrades by the factor F_{Upgrade}

Under both previous conditions, an upgrade interval for the link is subdivided into p_{jk} equal smaller intervals, such that the utilization is reduced p_{jk} times by the factor $F_{\text{Upgrade, TE}}$ instead of one reduction by F_{Upgrade}. Figure 1 includes an example for $p_{jk} = 2$ where an upgrade interval for $F_{\text{Upgrade}} = 4$ is divided into 2 subintervals with $F_{\text{Upgrade}} = 2$. For a proof that the previous conditions maximize the obtainable mean utilization, it can be shown by differentiation, that a global maximum is reached for arbitrary, non-equidistant upgrade time points within an interval I_{Upgrade} as well as corresponding non-equal factors for each upgrade. Thus in the optimum scenario TE can improve the utilisation up to

$$\overline{u}_{jk} = T_{\text{Upgrade}} \frac{1-1/\sqrt[p_{jk}]{F_{\text{Upgrade}}}}{\ln(F_{\text{Upgrade}})/p_{jk}}.$$

The corresponding upper bound on the TE gain is

$$g_{TE} \le p_{jk} \frac{1-1/\sqrt[p_{jk}]{F_{\text{Upgrade}}}}{1-1/F_{\text{Upgrade}}}. \qquad (3)$$

Evaluation of (3) in a network with a connectivity degree of e.g. 4, such that at least 4 paths over disjoint links exist between each pair of nodes and thus $p_{jk} \ge 4$, shows that the TE gain over an upgrade period is bounded by $g_{TE} \le 1.273$ for $F_{\text{Upgrade}} = 2$ and $g_{TE} \le 1.563$ for $F_{\text{Upgrade}} = 4$. As a consequence, TE allows to keep the utilization in a corridor much closer to the threshold than SPF. Since the optimum conditions are usually not perfectly achieved, we have investigated the efficiency of TE in realistic case studies with an example shown in section 4.

SUMMARY OF TE ALGORITHMS

The LSP design for load balancing has to be optimised in order to minimize the maximum load on a link in the network with predefined capacities as constraints. Since quality of service properties mainly depend on the link loads and are affected by non-predictable fluctuations of traffic demands leading to temporary overload situations, this optimisation goal aims at minimizing QoS degradation with regard to previously measured traffic demands. The implications of the utilisation level on packet loss and delay as main performance characteristics of switching systems has been investigated in [10] for IP backbone networks. Methods to estimate the traffic matrix are proposed in [3][11][13].

We have implemented linear programming and a heuristic optimisation method using simulated annealing in order to balance the load distribution [12][15]. Linear programming can be used to determine the maximum admissible traffic load $f_{\mathrm{TE}}T$ if the traffic demands can be split up over arbitrary paths, which is not possible or desirable in IP networks. But for that case an exact calculation of the optimum LSP design is possible at a reasonable computational complexity. When each traffic demand has to be carried on a unique path, then the optimisation can be shown to be in the class of NP hard problems [1][2][4][9], which has be handled by heuristic solutions. Instead of allocating traffic on explicit paths, TE approaches have been proposed by manipulating link weights in order to balance the network load [7]. Since the number of links and thus the number of variables to control the traffic flows in this way is essentially smaller than the number of explicit paths, only sub-optimum solutions can be expected, but the approach is applicable in pure IP networks without support by MPLS.

The linear programming algorithm introduces variables to represent the flows on the links distinguished by each originating source. The variables are related by equations expressing flow conservation at each node, such that the rate of incoming traffic equals the rate of departing traffic including external flows entering at a source and leaving the network at their destinations. The capacity of each link represents a constraint for the sum of allocated traffic rates. For the resulting set of linear boundary equations the simplex algorithm is available as a standard method in operations research to obtain an exact optimum solution [8].

The linear programming method offers some flexibility with regard to the considered cost function and optimisation goal, but heuristic algorithms are a more flexible and scalable means when approximate solutions are sufficient. We apply a heuristic method under the assumption that each demand in the traffic matrix has to be delivered on a unique path from source to destination, which represents an additional constraint. An explicit search for paths is carried out that minimize weighted costs on the links in a network-wide view. The cost function is increasing with the load on each link, leading to a tendency to shift paths away from highly utilized links. The path for traffic demands are assigned sequentially where the largest demand first principle is an appropriate initial choice as known for comparable problems, e.g. bin packing. Thereafter simulated annealing [12][15] is used for stepwise improvement of the solution. Our experience shows that the TE gain obtained by simulated annealing is usually close to the linear programming solution and both approaches show no essential differences when applied in the following case study.

CASE STUDY OF UTILIZATION DURING A CAPACITY UPGRADE PROCESS

In an example we consider the network topology shown in figure 3. In detail, the network is completely doubled with two subsets of nodes $\aleph_1 = \{A_1, B_1, \ldots, J_1\}$ and $\aleph_2 = \{A_2, B_2, \ldots, J_2\}$ each of which is connected in the given structure, as illustrated for D, where a link between each pair of nodes (A_1, A_2), (B_1, B_2) ,…, (J_1, J_2) is also included. Initially we assume that all connections are bidirectional with a unique capacity of all links in both directions.

The traffic matrix is composed of contributions for uniform demands between pairs of nodes in each subset as well as traffic originating from source nodes E_1, E_2 and J_1. This may correspond to external traffic via global peering points entering the network at a subset of nodes. In particular, the traffic matrix $T = (t_{jk})$ is described by

$$t_{jk} = t_{jk}^{(1)} + t_{jk}^{(2)} + t_{jk}^{(E1)} + t_{jk}^{(E2)} + t_{jk}^{(J1)}$$

$$t_{jk}^{(1)} = \begin{cases} 1 & \text{if } j,k \in \aleph_1;\ j \neq k; \\ 0 & \text{else;} \end{cases}$$

$$t_{jk}^{(2)} = \begin{cases} 1 & \text{if } j,k \in \aleph_2;\ j \neq k; \\ 0 & \text{else;} \end{cases}$$

$$t_{jk}^{(E1)} = \begin{cases} 100 f_k^{E1}/27 & \text{if } j = E_1;\ j \neq k; \\ 20 f_j^{E1}/27 & \text{if } k = E_1;\ j \neq k; \end{cases}$$

$$t_{jk}^{(E2)} = \begin{cases} 100 f_k^{E2}/23 & \text{if } j = E_2;\ j \neq k; \\ 20 f_j^{E2}/23 & \text{if } k = E_2;\ j \neq k; \end{cases}$$

$$t_{jk}^{(J1)} = \begin{cases} 5 f_k^{J1}/3 & \text{if } j = J_1;\ j \neq k; \\ f_j^{J1}/3 & \text{if } k = J_1;\ j \neq k. \end{cases}$$

The latter portions of traffic from and to E_1, E_2 and J_1 has a non-uniform distribution over the other nodes, which is specified in tables 1 and 2:

Node k:	A_1	B_1	C_1	D_1	E_1	F_1	G_1	H_1	I_1	J_1
f_k^{E1}	3	1	2	1	0	2	1	1	1	1
f_k^{E2}	1	2	2	1	2	1	1	4	1	1
f_k^{J1}	2	1	2	1	2	1	1	1	0	1

Table 1: External source traffic distribution in $\aleph_1$

Node k:	A_2	B_2	C_2	D_2	E_2	F_2	G_2	H_2	I_2	J_2
f_k^{E1}	1	1	2	1	2	1	1	1	3	1
f_k^{E2}	1	1	2	1	0	1	1	3	2	1
f_k^{J1}	1	1	3	1	2	1	1	2	2	2

Table 2: External source traffic distribution in $\aleph_2$

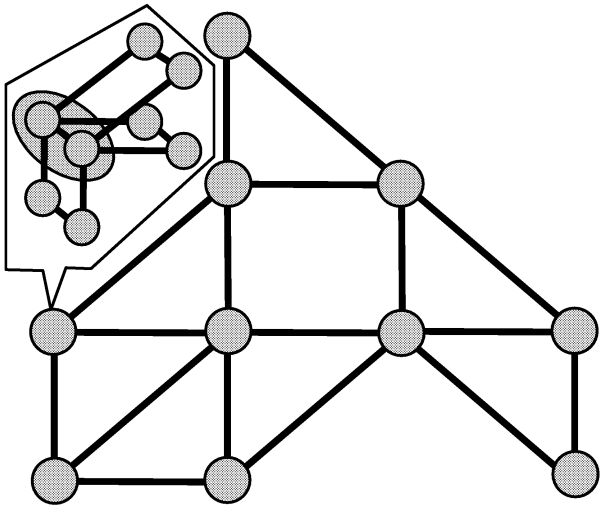


Figure 3: Network topology of the example

The network topology and traffic matrix have been chosen at random but in order to reflect the usual scenario in service provider networks with a portion of uniform traffic between nodes superposed by non-uniform traffic from external sources, which is concentrated on some of the nodes.

For modelling the process of capacity adaptation to increasing traffic demands we started with the previously described topology, traffic matrix and an initial utilization level as indicated in figures 4 and 5. We assume a constant increase of traffic by a factor 2.5 per year $t_{jk}(x)) = t_{jk}(0))\ \exp(\omega x)$ and a corresponding rate per week. Link upgrades are carried out on a weekly time scale whenever the link utilization exceeds a threshold of 50%. This process is traced over three years or 156 weeks with comparison of traffic allocation by SPF and by TE.

In case of TE, a reoptimization is immediately carried out each time when a link has been upgraded. The resulting mean utilization of network links is shown for upgrades on double capacity in figure 4 and to 4-fold capacity in figure 5. When traffic engineering is applied, then the utilization threshold is often exceeded by a number of links, which together form a bottleneck or a minimum cut through the network.

Therefore we included two variants, where the complete bottleneck is upgraded in a single step or only one of the links on a bottleneck is upgraded, respectively. As expected, the latter case leads to higher mean utilization as illustrated in figures 4 and 5. It is also visible in the figures that the utilization tends to the values predicted by the analysis of section 2 from an initially lower level. The TE support does not exploit the maximum, which is close to the threshold, but yields an essential improvement as compared to SPF.

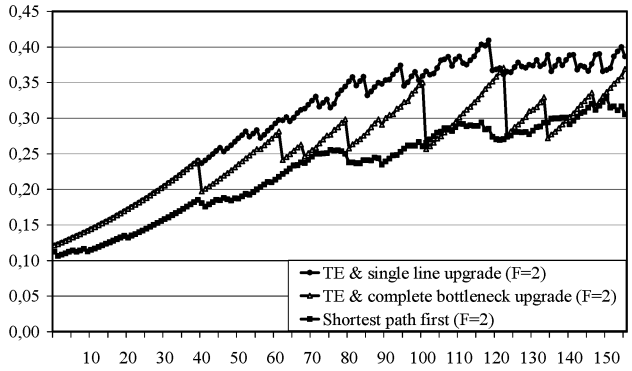


Figure 4: Network wide mean utilization

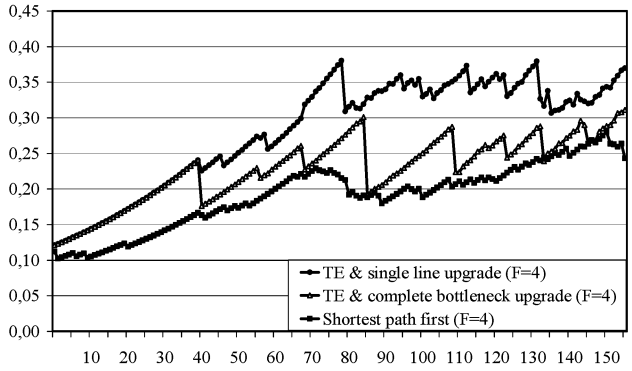


Figure 5: Network wide mean utilization

The previous comparison of the influence of TE and SPF on the link utilization may be inappropriate due to the fact that TE may increase utilization by allocating traffic on longer paths. Although optimum LSP designs are experienced to include only a small set of paths which deviate from the shortest possible path, we consider the ratio of the complete allocated traffic flow to the total installed capacity on the links as a fair measure to compare TE to SPF. The results are shown in figure 6 for upgrade steps by factor 2 and in figure 7 for upgrade steps by factor 4.

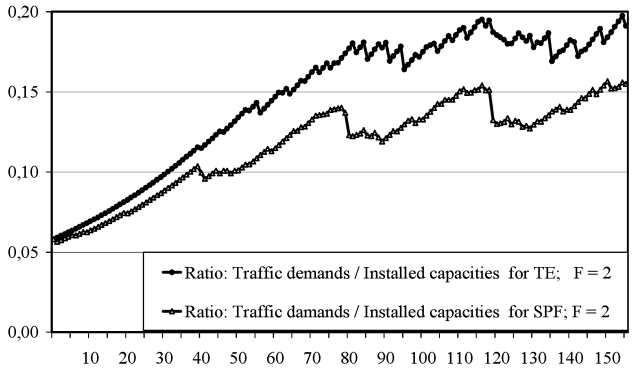


Figure 6: Ratio of flow assignment to capacity

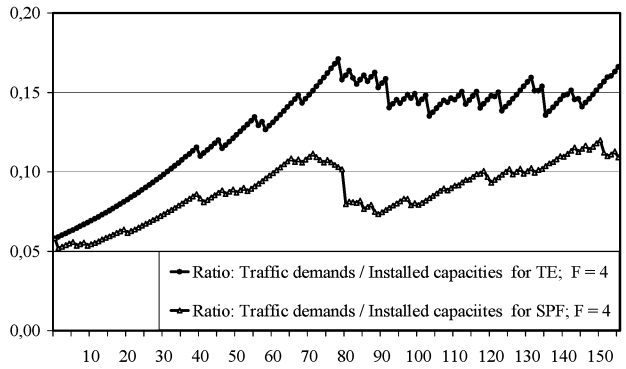


Figure 7: Ratio of flow assignment to capacity

Finally, figure 8 shows the TE gain expressed by the ratio of total installed capacity with SPF as related to TE. The example shows that the TE gain can temporarily exceed the bound, which has been derived for a single link over a complete upgrade period and which is also valid in a long term steady state view.

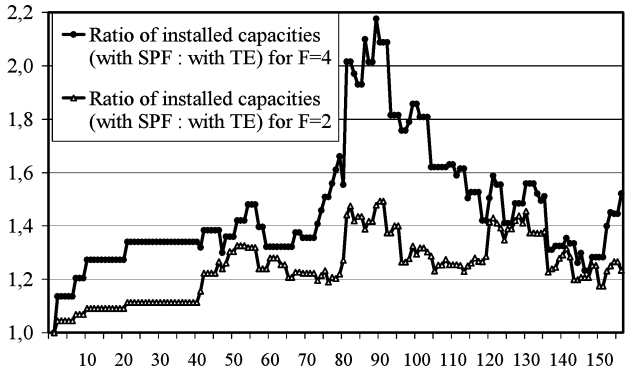


Figure 8: Ratio of capacities required for SPF and TE

Note, that the results are independent of the increase rate ω and still apply to the case of a time dependent non-negative increase rate $\omega(x)$ and even for different increase $\omega_{jk}(x)$ of each demand over time

$$t_{jk}(x) = t_{jk}(0)\,\exp(\int \omega_{jk}(t)\,dt)\,,$$

provided that the increase develops independent of the current phases of link extension.

CONCLUSION AND OUTLOOK

This study of traffic engineering in IP networks with focus on the requirement for a steady scaling of resources according to increasing traffic demands shows that TE can significantly improve the utilization by filling up gaps arising in SPF routing after each link upgrade. Analytical bounds on this effect are compared to the performance under usual conditions. In this way the benefit of TE is captured in normal operation without regarding short term traffic fluctuations or failure situations, where TE provides additional benefit by the opportunity of a flexible response by redistributing traffic.

In current investigations we are also concerned with the relation of TE to QoS aspects. Previously, we assumed the same upgrade thresholds to be applied in the SPF and TE cases. This threshold is decisive for the occurrence of overload situations as the main driver of QoS degradation. The evaluation of QoS aspects requires stochastic modelling of the time varying behaviour of traffic demands as well as for random delay between the planning for link upgrades and their execution time. The main focus of an extended modelling approach is to determine the upgrade threshold of link utilization appropriately to meet reliability constraints at a minimum of investigation into installed resources.

REFERENCES

[1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, Network flows: Theory, algorithms and applications, Prentice Hall (1993)

[2] V.K. Balakrishnan, Network optimization, Chapman & Hall, London (1995)

[3] N. Benameur and J.W. Roberts, Traffic matrix interference in IP networks, Proc. NETWORKS, Munich (2002) 151-156

[4] D. Bienstock, New developments in solving multicommodity flow problems, Proc. 6. INFORMS Telecom. Conf. TC4.1, Boca Raton (2002)

[5] A. Elwalid, C. Lin, S. Low and I. Widjaja, MATE: MPLS Adaptive Traffic Engineering, Proc. IEEE INFOCOM (2001)

[6] A. Feldmann, A. Greenberg, C. Lund, N. Reingold and J. Rexford, NetScope: Traffic engineering for IP networks, IEEE Network, special issue on Internet traffic engineering, Ed. Z. Wang (March 2000) 11-19

[7] B. Fortz and M. Thorup, Internet traffic engineering by optimizing OSPF weights, Proc. IEEE INFOCOM (2000)

[8] M. Franzke and A. Pönitz, Global shortest path solutions for the traffic engineering problem, Proc. NETWORKS, Munich (2002) 275-278

[9] G. Hasslinger and S. Schnitter, Optimized load distribution in MPLS networks, Telecommunications Network Design and Management, Kluwer Akad. Publ. (2002) 125-142

[10] G. Hasslinger, Quality-of-service analysis for statistical multiplexing with Gaussian distributed and autoregressive input modeling, Telecommunication Systems 16 (2001) 315-336

[11] G. Hasslinger, A decomposition method to estimate the traffic demand matrix from link measurement, Proc. SoftCOM, 10. Internat. Conf. on Software, Telecommunications and Computer Networks, Split, Croatia (2002) 695-699

[12] M. Kolonko, Some new results on simulated annealing applied to the job shop scheduling problem, Europ. J. Operational Research, 113 (1999) 123-136

[13] A. Medina, C. Fraleight, N. Taft, S. Bhattacharyya and C. Diot, Taxonomy of IP traffic matrices, Proc. SPIE, Vol. 4868, Boston (2002) 200-211

[14] E. Rosen, A. Viswanathan and R. Callon, Multiprotocol label switching architecture, RFC 3031, Proposed IETF standard (2001)

[15] S. Schnitter and G. Hasslinger, Heuristic Solutions to the LSP-design for MPLS Traffic Engineering, Proc. NETWORKS 2002, Munich (2002) 269-273

[16] R.D. Wollmer, Optimal investment in capacity expansion of arcs in stochastic multicommodity network flows Proc. 6. INFORMS Telecom. Conf. WA1.4, Boca Raton (2002)

SIMULATION WITH PETRI NETS

PETRI NETS
I

Switched LAN Simulation by Colored Petri Nets

Dmitry A. Zaitsev
Department of Communication Networks,
Odessa National Telecommunication Academy,
Kuznechnaya, 1, Odessa, 65029, Ukraine
E-mail: zaitsev_dmitry@mail.ru

KEYWORDS
Computer networks, Switch, Petri net, Estimation.

ABSTRACT

The methodology of switched LAN models construction in the form of colored Petri net is introduced. For the simulation and analysis of the model the Design / CPN tool is used. The tasks of estimation of LAN switch buffer size and network response time were solved. The components of the model are switches, servers and workstations.

INTRODUCTION

In work (Hunt 1999) it is stated that the technology of switching is prospective for bandwidth increase in local and global computer networks. In article (Elsaadany et al. 1995) the switched network is investigated by means of stochastic nets with queue; the influence of switch buffer size on the quantity of the lost packets and the general productivity of the network is considered. Distinct from pointed models, Petri nets contain facilities for more precise description of network architecture and traffic and allow representation of the interaction within the client-server systems.

For the description of the real world objects it is not usual to apply basic Petri nets (Peterson 1981). Its play the central role in theoretical investigations. For practical purposes various extended models such as colored, timed and hierarchical nets are used. The theory of the colored Petri nets was developed in monograph (Jensen 1997), while a formal description of timed nets behavior is the subject of article (Zaitsev and Sleptsov 1998).

In present work the switched local area network investigation is implemented by means of simulation system Design/CPN (Albert et al. 1989) and the task of network response time estimation was solved.

DESCRIPTION OF RESEARCHED OBJECT

The base element of the switched Local Area Network (LAN) Ethernet (IEEE 803.x) is the switch of frames. Logically the switch is constituted by the set of ports (Rahul 2002). The LAN segment (for example, made up via hub) or the terminal equipment such as a workstation or server may be attached to each port. The task of the switch is the forwarding of incoming frame to the port that the target device is connected to. The usage of the switch allows for a decrease in the quantity of collisions so the frame is transmitted only to the target port and results in an increase bandwidth. Moreover the quality of information protection rises with a reduction of ability to overhear traffic. The scheme of small office switched network is presented in Fig. 1.

To determine the target port number for the incoming frame a static or dynamic switching table is used. This table contains the port number for each known Media Access Control (MAC) address. Algorithms of dynamic table maintenance are based on traffic listening for the search of unknown source MAC addresses and the creation of new records for such addresses. During the processing of unknown destination address the frame is transmitted to all the switch ports.

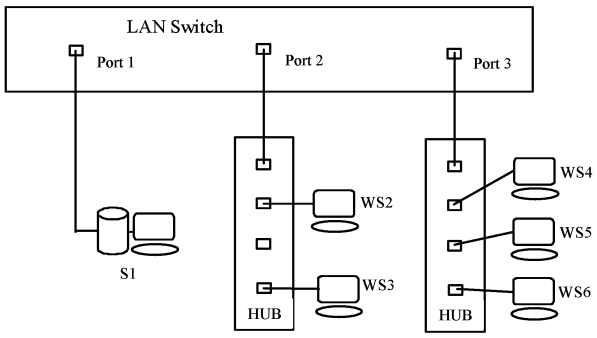


Fig. 1. Scheme of small office switched LAN.

MODEL OF LAN SWITCH

Let us construct the model for a given static switching table. We shall consider separate input and output frame buffers for each port and a common buffer of the switched frames. A model of the switch is presented in Fig. 2. Hosts disposition according to Fig. 1 was used for the testing of the model.

MAC address of the host is represented by the integer number. Moreover, content of the frame is not considered. Data type *frm* describes the frames of the network, data type *swch* represents the switching table records, and data type *swchfrm* describes the switched frames waiting for output buffer allocation. Places *PortX In* and *PortX Out* represent input and output buffers of port *X* accordingly. Place *SwitchTable* models the switching table; each token in this place represents the record of the switching table. Place *Buffer* corresponds to the switched frames' buffer. Transitions *InX* model the processing of input frames. The frame is extracting from the input buffer only in cases where the switching table contains a record with an address

that equals the destination address of the frame; during the frame displacement the target port number is stored in the buffer. Transitions *OutX* model the displacement of switched frames to output ports' buffers. Fixed time delays are assigned to the operations of the switching and the writing of the frame to the output buffer.

MODELS OF WORKSTATIONS AND SERVERS

To investigate the frames' flow transmitting through the local area network and to estimate the network response time it is necessary to supply the model constructed with the models of terminal devices attached to the network. The general model assembling may be provided by means of union (fusion) of places.

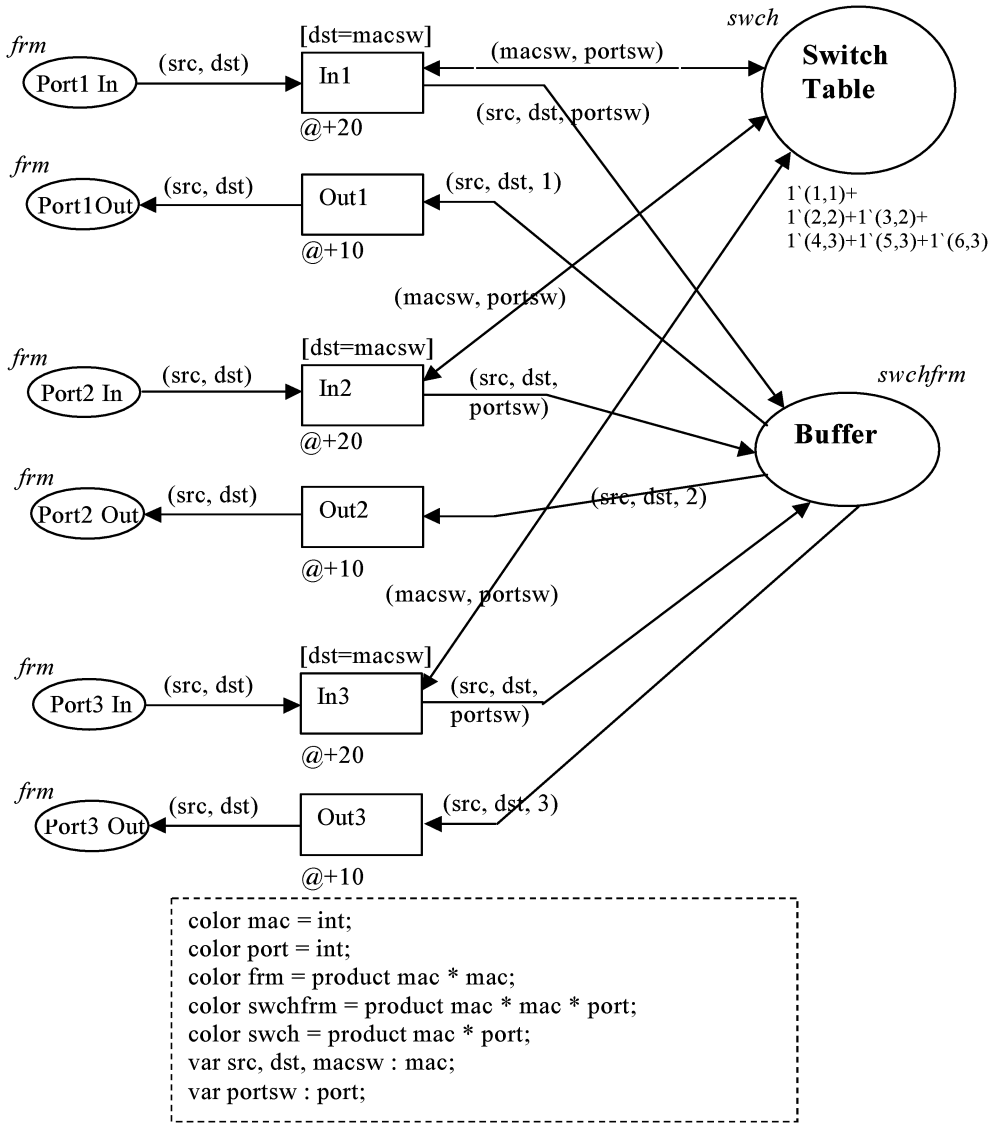
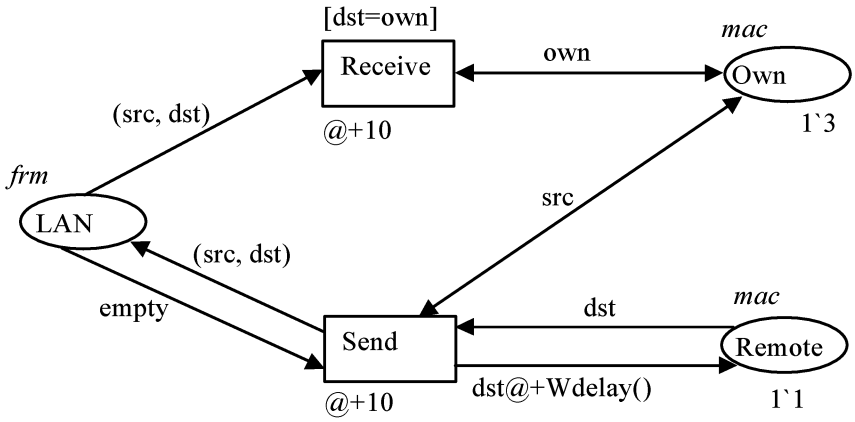


Fig. 2. Model of LAN switch.

On the peculiarity of the traffic's form we shall separate workstations and servers. For an accepted degree of elaboration we shall consider the periodically repeated requests of the workstations to the servers with the random uniform distributed delays. On reply to accepted request the server sends a few packets to the address of the requested workstation. The number of packets sent and the time delays are the uniform distributed random values.



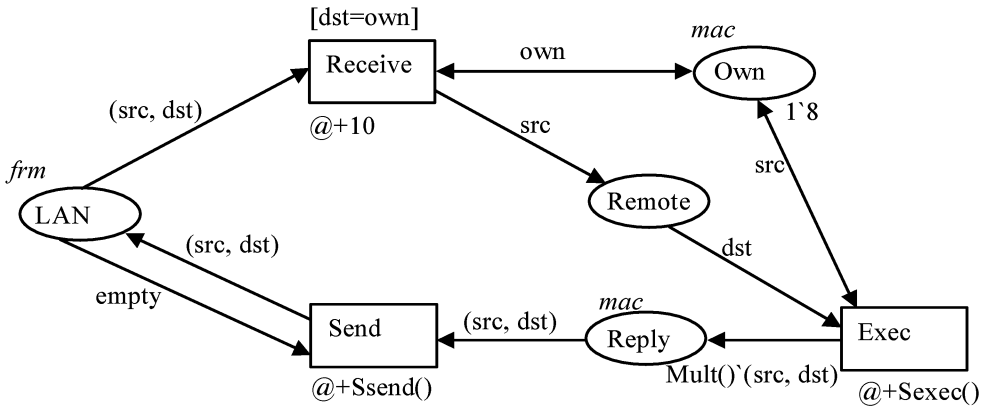

```
color SendDelay = int with 1000..10000 declare ran;
fun Wdelay = ran`SendDelay;
```

Fig. 3. Model of workstation.

A model of workstation is represented in Fig. 3. Place *LAN* models the segment of the local area network that the workstation is attached to. The workstation listens to the network by means of a transition *Receive* that receives frames with the destination address that equals the own address of the workstation saved in the place *Own*. The processing of received frames is represented by the simple absorption of them. The workstation sends periodic requests to the server by means of transition *Send*. The servers' addresses are held in the place *Remote*. After sending request the usage of the server's address is locked by the random time delay given by the function *Wdelay()*. The sending of the frame is implemented only if the LAN segment is free. It operates by checking of the place *LAN* for the lack of tokens. In such a manner we may interact with a few servers holding their addresses in the place *Remote*.

A model of the server is represented in Fig. 4. The listening of the network is similar to the model of the workstation but is distinct in that the frame source address is held in the place *Remote*. Transition *Exec* models the execution of the workstation's request by the server. As a result of the execution request the server generates a random number of the response frames that are held in the place *Reply*. Then these frames are transmitted into the network by the transition *Send*.

The assembly of the general local area network model is implemented by the union of the places *LAN* of workstations and servers for each of the segments. The model of the switch is attached to the models of segments by means of additional transitions *ReceiveX* and *SendX* accepting frames into the input buffer and transmitting frames from the output buffer accordingly for each switch port. The developed model is analysed in interactive and automatic modes of the simulation system Design/CPN. The interval of the model time between sending of the request token and receiving the answer tokens is the estimated network response time.




```
color SendDelay = int with 100..300 declare ran;
color ExecDelay = int with 1000..2000 declare ran;
color MultReply = int with 10..100 declare ran;
fun Ssend = ran`SendDelay;
fun Sexec = ran`ExecDelay;
fun Mult = ran`MultReply;
```

Fig. 4. Model of server.

Hence, in the present work the technology of the switched local area networks' models development is represented. Also, the principles of the estimation of network response time with the help of the simulation system Design/CPN is described. Colored Petri nets usage allows the peculiarity of interaction within the client-server systems to be taken into account.

REFERENCES

Albert K., Jensen K., Shapiro R. 1989. "Design/CPN: A Tool Package Supporting the Use of Colored Nets". *Petri Net Newsletter*, April, 22-35.

Elsaadany A., Singhal M., Ming T. Lui. 1995. "Performance study of buffering within switches in local area networks". In *Proc. of 4th International Conference on Computer Communications and Networks*, 451-452.

Jensen K. 1997. Colored Petri Nets – Basic Concepts, Analysis Methods and Practical Use. Vol. 1-3, *Springer-Verlag*.

Hunt R. 1999. "Evolving Technologies for New Internet Applications". *IEEE Internet Computing*, No. 5, 16-26.

Peterson J. 1981. Petri Net Theory and the Modelling of Systems, *Prentice Hall*.

Rahul V. 2002. LAN Switching. *OHIO*.

Zaitsev D.A., Sleptsov A.I. 1998. "State Equations and Equivalent Transformations of Timed Petri Nets". *Cybernetics and System Analysis*, Vol. 33. No. 5, 659-672.

BIOGRAPHY



DMITRY ZAITSEV was born in USSR and in 1986 passed the Donetsk Polytechnic Institute on Applied Mathematics. In 1991 he obtained Ph.D. degree in the Automated Control from the Kiev Cybernetics Institute. He worked in Donetsk Polytechnic Institute, Surgut State University and also was a chief programmer in Nejron-Metaassembler, Opera-Topaz, Nevod and other projects. For example, Opera-Topaz is enterprise CAM software based on Petri net models. It is used at military plant Topaz in the production of the wide known radio spy station "Kolchuga".

Home page: http://www.geocities.com/zsoftua

REAL-TIME MODEL FOR PROCESSING BOTH SCHEDULING AND PETRI NETS APPROACHES

Mahfoud Mabed and Marc Bourcerie
LISA (FRE 2656 CNRS-Université d'Angers)
Department of Electrical Engineering
IUT, 4, Boulevard Lavoisier, BP 2018 F- 49016 Angers Cedex
France
Email: mahfoud.mabed@univ-angers.fr

KEY WORDS
Modeling, Discrete event systems, Petri Nets, Scheduling, Z/pZ Petri nets, Simulation.

ABSTRACT

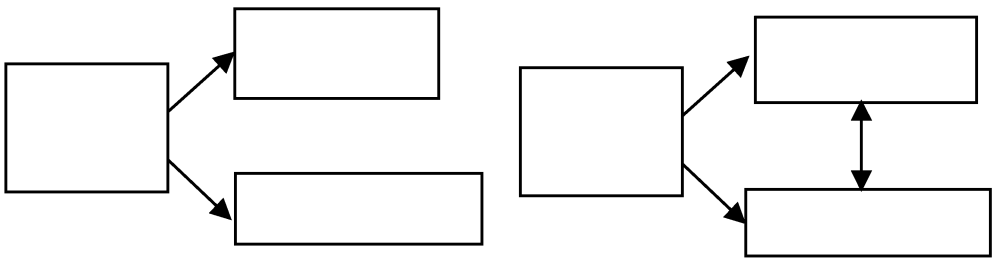
The notion of scheduling is a subject that characterizes many contemporary processes. Nevertheless, Petri nets do not represent a tool to realize a complex sequencing. The originality of this study is mainly the replacement of the structure, associated with colors, by a structure of field Z/pZ that allows the processing of colors succession with mathematics.
We propose in this paper to build a dynamic link between a scheduling solution and a simulation approach. We are interested here in modeling an electroplating line (mono-products) which involves very complicated sequences, modeled with difficulty by existing Timed Petri Nets (TPN) or Timed Colored Petri Nets (TCPN). A new kind of Petri nets called Z/pZ TPN is used based on a polynomial structure (Marteau et al 1995).

INTRODUCTION

Petri nets (PN) are used to model parallel or sequential processes. They have been applied to several types of systems (Baptiste and Legeard 1993). Their practical use for discrete event systems demonstrated a need for more powerful tools. Timed Colored Petri Nets (TCPN) are a significant improvement in this respect. However, they still show some inadequacies in modeling complex sequencing.
The idea of complex sequencing is strongly related to scheduling and real time systems. It appears when a resource has to be shared by more than one user or when a job must be handled by some concurrent entities. Although the contribution brought by the TCPN is considerable, the structure of set associated with colored tokens is not flexible enough. In order to succeed in the optimizing of some problems, it is often absolutely necessary to use a model, submitted to some constraints. Classical PN are a graphical tool, which is a disadvantage for modeling and analyzing complex systems. The functions attached to arcs in CP-nets sometimes seem to be more difficult to understand. We propose here to use Z/pZ Petri nets to model some complex sequencing. A mathematical structure is defined to represent the relation between the colors. This algebraic structure is an isomorphism between the set of colors and a finite field Z/pZ= IFp.
This field allows symbolic calculation using polynomials associated with arcs instead of linear functions. Using polynomial functions to describe complex sequencing in TCPN seems to be the best compromise between the power of mathematics and the understanding.

This paper addresses the modeling of an electroplating line with Z/pZ TPN. It illustrates that these nets can be easily applied to various industrial problems. Modifying a sequence leads to change the Z/pZ TPN parameters (i.e. the polynomials) while the net structure remains the same, which enables the dynamical sequence modification.
Even if the polynomials introduced into this presentation represent a clear evolution of the standard functions used in traditional TCPN, a characteristic does remain specific to all these polynomials. It consists of the definition of the color set. In Z/pZ TPN, there is only a global set of colors and all the polynomials take its values in it. This is why a circumstantial readjustment is always possible.
A new approach based on max-plus algebra theory has previously developed and led to propose solutions to the hoist scheduling problem. The results obtained represent a sequence, which is modeled in a Z/pZ Petri nets by polynomials. The idea is to build a dynamic link between resolution and simulation of such scheduling. We are interested in modeling a sequence in order to simulate it.

Before using Z/pZ TPN **With Z/pZ structure**



Figures 1: Global representation of the research

ALGEBRAIC STRUCTURE OF THE COLOR SET.

This section defines an algebraic structure enabling the modeling of sequential processes. This method uses algebraic operators (+, ×) which have an effect on the colors. We define a finite ring structure with p distinct elements {0, 1, 2, …, p-1} isomorphic with Z/pZ, a mathematical structure already used in areas close to discrete event systems (Mabed 2002). On this set, we define a polynomial function as an automorphism of Z/pZ.

$$p(x) = \sum_{i=0}^{p-1} a_i x^i$$

Simple sequence

Let us consider the sequence S:

$$S=(S_0,S_1,\ldots,S_{(p-1)})$$

We can obtain the corresponding polynomial P(X). A polynomial in Z/pZ[X] is completely given by its coefficients A_0 to $A_{(p-1)}$. We prove that a matrix $[L^p]$ exists there, making the correspondence between the p elements of S, describing the successors of p colors and p other elements, being the coefficients of P(X), i.e.

${}^t[A_0, A_1, \ldots, A_{(p-1)}]=[L^p].{}^t[S_0\ S_1 \ldots S_{(p-1)}]$.

$[L_k^p(X)]= [1\ X\ X^2 \ldots X^{(p-1)}].\ [L^p].\ [I_k]$, where I^k is a column matrix whose elements are equal to 1 in k^{th} row and equal to 0 elsewhere.

$[L_k^p(X)]$ is a (p-1) degree polynomial verifying the conditions :

$[L_k^p(k)]=1$ and $[L_k^p(X)]= 0,\ \forall\ X\neq k$

This polynomial is the interpolation Lagrange polynomial, which can be written, if considering the first theorem of Fermat : $[L_k^p(X)]= 1+(p-1)(X+p-k)^{(p-1)}$

when we develop $(X+p-k)^{(p-1)}$ by Newton's binomial in Z/pZ, we obtain $[L^p]$ matrix :

$$[L^p]= \begin{matrix} 1 & 0 & 0 & \ldots & 0 \\ 0 & (p-1)^{(p-2)} & (p-2)^{(p-2)} & \ldots & 1 \\ 0 & (p-1)(p-1)^{(p-3)} & (p-1)(p-2)^{(p-3)} & \ldots & p-1 \\ 0 & (p-1)^{(p-4)} & (p-2)^{(p-4)} & \ldots & 1 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ p-1 & p-1 & p-1 & \ldots & p-1 \end{matrix}$$

Let us apply this method to the sequencing vector S=(0,6,1,4,2,5,3), for p=7 the matrix $[L^7]$ is :

$$[L^7]= \begin{matrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 3 & 2 & 5 & 4 & 1 \\ 0 & 6 & 5 & 3 & 3 & 5 & 6 \\ 0 & 6 & 6 & 1 & 6 & 1 & 1 \\ 0 & 6 & 3 & 5 & 5 & 3 & 6 \\ 0 & 6 & 5 & 4 & 3 & 2 & 1 \\ 6 & 6 & 6 & 6 & 6 & 6 & 6 \end{matrix}$$

$P(X)= [1\ X\ X^2 \ldots X^6].[L^7].[F]$

with $[F]={}^t[P(0)\ P(1)\ P(2)\ P(3)\ P(4)\ P(5)\ P(6)]$ thus equal to ${}^t[6\ 4\ 5\ 0\ 2\ 3\ 1]$.

we obtain $P(X)=6+6.X+6.X^2+X^4+6.X^5$

and easily verify that P(0)=6, P(6)=1, P(1)=4, P(4)=2, P(2)=5, P(5)=3, P(3)=0;

As said before if the length of the sequence does not correspond to a prime number (n<p), we go back to the previous case and consider (p-m) fictitious tasks that will not be used.

Complex sequence

A complex sequence means a sequence in which a task can have several distinct successors, for instance $(\ldots,o,o_1,\ldots,o,o_2\ldots)$. we use (x, y) couples in which x is a task index and y indicates how many time this task appears in the sequence. We will obtain $((o_1,0), (o_2,0), (o_1,1),\ldots)$.

Let us consider now the sequence S=(0,1,2,1,3,1), where the task (color(1)) has three distinct successors. Each color can be written in Z/4Z × Z/3Z as a couple (x, y) where y is the number of occurrence. The sequence becomes:

S = ((0,0), (1,0), (2,0), (1,1), (3,0), (1,2)

When the number n of colors is not prime, Z/nZ is not a field and the problem does not have a solution each time, since any element of a ring does not have an inverse. We must consider a field Z/pZ with p prime and $p \geq \max(m,n)$; We can prove that the problem always has a solution, i.e. there is always a polynomial for any sequence, provided the number of colors be prime.

The number of colors used is m = 4, the nearest prime number $p \geq \max(4,3)$ is equal to 5. The sequence will be represented by P(X,Y) polynomial in Z/5Z.

After calculus [10,11] the polynomial P(X,Y) , will be written :

$P(X,Y) = (P_1(X,Y), P_2(X,Y))$ on Z/5Z:

$P_1(X,Y)=3X^3+3X+3Y^2+3Y+1$;

$P_2(X,Y)=4X^3+X^2$

We can easily verify

$P(1,1) = (P_1(1,1), P_2(1,1))$:

- $P_1(1,1) = 3 + 3 + 3 + 3 + 1 = 13 \Rightarrow P_1(1,1) = 3$; in Z/5Z
- $P_2(1,1) = 4+1 = 5 \Rightarrow P_2(1,1) = 0$ in Z/5Z

P (1,1) = (3,0) ;

P (1,0) = (7,0) = (2,0) in Z/5Z; and so on

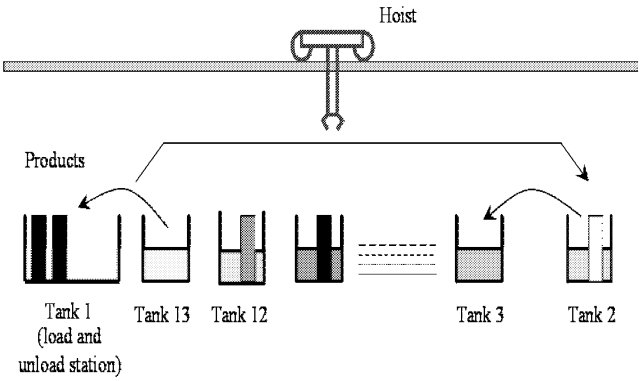
Whether sequences are complete or incomplete, complex or simple, the calculation of such polynomials always remains possible (Mabed 2002) (Marteau and Bourcerie 1996).

Each polynomial characterizes a well defined sequence in a known color set E. Each evolution or disturbance, which can happens, will leave this set unchanged. Only the parameters are to be changed.

APPLICATION

Production line fed with one or more hoists are used in a large number of cyclic industrial applications. The moving

devices (hoists) transport products from one station to another, according to a sequence order (Bloch 1999). This problem is characterized by the nature of the processing times, which are variables of the model. Each of them has to take a value in a given interval $[t_i, \delta_i]$. The result of a schedule is firstly the sequence of the movements and secondly the values of all processing times (mabed et al 2001a) (see Figures 2). We consider a production line with 13 different stations and one hoist. These stations are arranged in a row and are fed with one hoist.

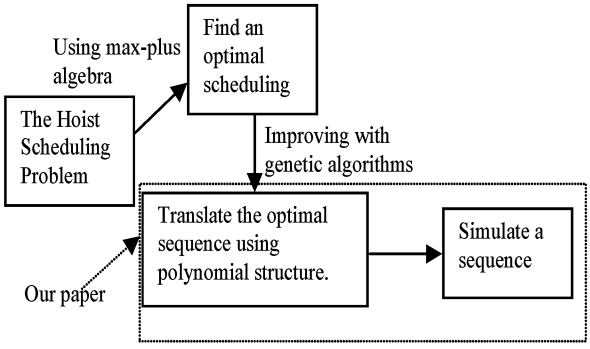


Figures 2: An Electroplating Line (1 Hoist, 13 tanks)

The first station represented by Tank(1), is a shared station for both loading and unloading purpose. A soak operation is performed in each other station. The hoist transfers a product from one tank to another, while respecting minimal and maximum duration.

The following assumptions will be respected throughout this presentation except otherwise specified:
– The processing time in each tank is bounded between a minimal and a maximum duration $[t_i, \delta_i]$: i = 0 to 12;
– Each tank can receive only one product at the same time;
– We are interested here about modeling an electroplating line. We admit that the hoist is always available.

Let us study the mono-products modeling case (one hoist, tanks with unit capacity, same processing for all the products, n products on the line).
Each product will be successively treated in several tanks and then deposed into the unloading station (tank(14)). There are two major statuses: product transfer from a tank to another and processing.



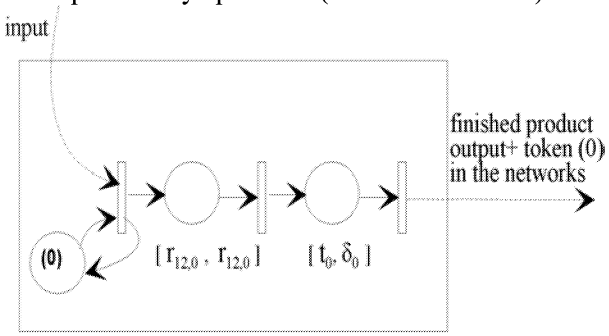
Figures 3: Paper objective

The sequence presented on Figures5 is :
1-2-3-…-12-13-1. It means that each product processed on the line follows this sequence. For better clearness, we represent, unload operation by a fictitious tank (14) (δ_1 = δ_{14} = inf (infinite)). Two labels are given with each place. The first one t, is the minimum time that a token entering the place must spend. The second label δ, is the maximum time that this token can spend in this place. Each tank has its own bounded time intervals. If a token enters a place of TCPN, it must spend a minimum time t_i in the place, and leave it at a maximum time δ_i at the latest. Each time a tank is emptied, it is immediately available for the next product on the line. The last operation of each product sequence is to unload it.

Z/PZ TIMED PETRI NETS (TPN) MODEL

Assumptions:
– The first tank (1) (token(0)) filling the unloading and loading tank, we will assume that there is no event before this.
– In this Petri Nets, tokens represent tanks. Each time a transition is fired, it will mean either a product is being transported from a tank to another, or that it is processed.

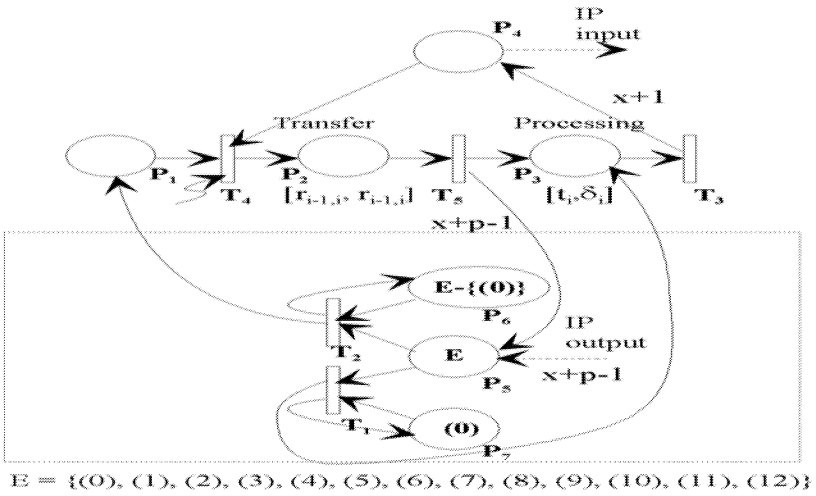
When we elaborate this TPN, several difficulties were encountered due to the two associated stations (loading, unloading) represented by tank(1). The first operation is to load the product, which is represented by a processing. The initial conditions lead us, to have one token in tank(1) for each new product on the line. Thus, we propose an Initialization Procedure (IP), which will help us to manage these constraints (see Figures 4). This procedure is necessary, as the products arrive directly at the load tank, without preliminary operations (Mabed and al 2001).



Figures 4: Initialization Procedure (IP)

It relates only to the products arriving at the end of their processing and then requires being unloaded. This procedure will be used for all the following models. However, we will have to make some modifications to the color set definition. Accessing to the (IP) procedure depends on the token (0) availability in the main Petri nets (see Figures 5). When, a product is completely unloaded, tank will become available for another use. Certainly, the obtained TCPN is adequate for this problem, but it remains rigid with probable extensions. (Mabed 2001b) (Mabed 2001a). The use of Z/pZ TPN, remains the best possibilities to represent complex systems. For each defined sequence, it can be written by a representative polynomial in Z/pZ. A complex system needs to be dynamically controlled. Where

(x+1) polynomial, constitutes a successor of x in the selected sequence. The allocation of polynomial functions on arcs of Z/pZ TPN is a good way to respect the constraints brought by notions such as sequencing and scheduling. Even, if the polynomials introduced into this presentation show a clear evolution of the standard functions used in traditional TCPN, a characteristic does remain specific to all these polynomials. It consists of the definition of the color set. In Z/pZ TPN, there is only a global color set and all the polynomials take its values in it. This is why a circumstantial readjustment is always possible. Whereas, the functions used in TCPN could fill the same requirements with different color set. This would give them a great rigidity at the risk management level. In practice, hoist assignments is carried out, by heuristic or stochastic methods.



Figures 5: Z/pZ TPN Model

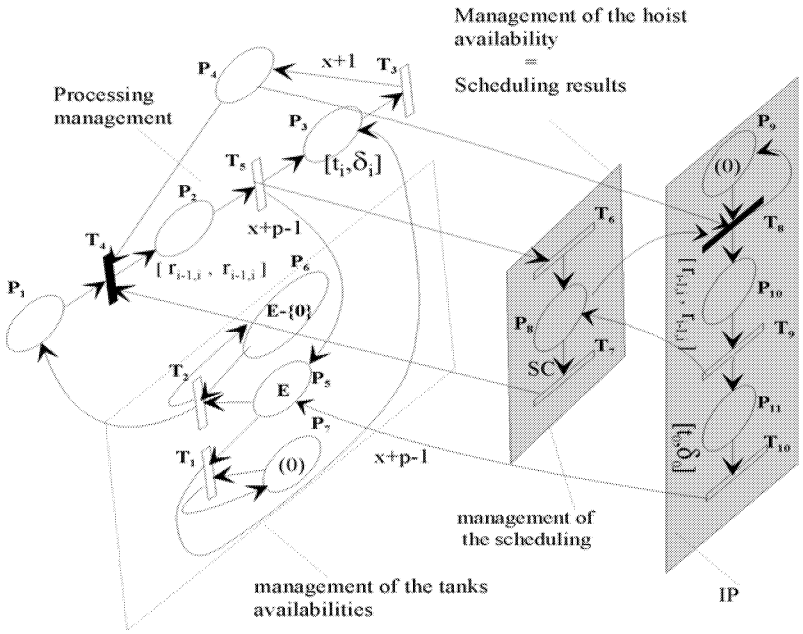
This section has presented an electroplating line modeled by Z/pZ TPN. It illustrates that these nets can be easily applied to various industrial problems. We find that Z/pZ TPN fills this spot successfully.

GLOBAL Z/PZ TPN REACTIVE MODEL

Petri nets have been traditionally employed in simulating approaches based on discrete event systems. The first goal of this work is to create a dynamic link between two fields:

- Scheduling approach: based on several stochastic or mathematical methods.
- Simulation approach based on Petri nets.

We present in Figures (6) a global model for the hoist scheduling problem. We have elaborated this model in two parts. The first one was for modeling the electroplating line. The second one for representing the results of the schedule approach (not developed in this paper). The sequence representing a schedule has been realized by the calculus of a polynomial. This polynomial characterizes a well defined schedule in a known color set. Each evolution or disturbance lead this set unchanged. Only the parameters are to be changed. The model presented below allows a dynamic management.



Figures 6: The HSP problem with Z/pZ TPN Model

CONCLUSIONS

This paper uses the Fp nets, which are Colored Petri nets having a finite field structure on the color set. The originality of this study is mainly the replacement of the structure, associated with colors, by a structure of field Z/pZ that allows the processing of colors succession with mathematics.

This mathematical structure allows an easy modeling of sequences by polynomials and symbolic calculation, thus enhancing the modeling capabilities of classical TCPN. Furthermore, this mathematical support may enable us to consider various problem of optimization, particularly those dealing with the resolution of scheduling problems, one of our points of interest. This new tool can be used as basis for further studies on scheduling. We already note that this method offers the advantage to operate on complex models, allows dynamic model evolutions of PN, and leads to predict interesting solutions in scheduling problems and real time systems. Indeed, it is not necessary to do the whole calculation, if the scheduling is modified. Developing these polynomials to evaluate a model, facilitates modeling any kind of sequence. We find that Z/pZ TPN fills this purpose successfully. It simplifies the net structure significantly so that makes the analysis process much easier.

REFERENCES

Baptiste, P.; and B. Legeard, Optimization with constraint logic programming : The Hoist scheduling problem solved with various solvers, 8th International Conference on Applications of Artificial Intelligence in Engineering, AIENG, Toulouse, France, 1993.

Bloch, C. Contribution à l'ordonnancement dynamique de lignes de traitement de surface, Thèse de Doctorat, Laboratoire d'Automatique de Besançon (UMR CNRS 6596), Université de Franche-Comté, 1999.

Borevitch, Z-I.; and I-R. Chafarevitch, Théorie des Nombres, (Gauthier-Villars), Paris, 1967.

Bourcerie, M.; and J-Y. Morel, Algebraically structured coloured Petri nets to model sequential processes. IEEE Transactions on Systems, Man and Cybernetics, Part B : Cybernetics, Vol. 27. n°4. 1997.

Di Mascolo, M. Modeling and analysis of kanban systems using Petri nets and queuing networks. In : Manufactured in the Netherlands (International Journal of Flexible Manufacturing Systems), p. 275-307. Kluwer Academic Publishers, Boston, 1991.

Mabed, M.; F. Bousseau and M. Bourcerie, Modeling dynamic complex systems : application to an electroplating line, Multiconference IEEE/IFAC/ IFIP, Second conference on Management and Control of production and logistics, MCPL2000, Grenoble, France, 2000.

Mabed, M.; F. Bousseau and M. Bourcerie, Modélisation d'une ligne de Galvanoplastie par les RdPT Z/pZ : cas Multi-produits, 3e Conférence Francophone de Modélisation et SIMulation, Conception, Analyse et Gestion des systèmes Industriels , MOSIM'01 Troyes, France, 2001.

Mabed, M.; F. Bousseau and M. Bourcerie, Z/pZ Petri nets model with application to an electroplating line.IEEE International Conference on Systems, Man and Cybernetics October 7-10, 2001, Tucson, Arizona, USA.

Mabed, M. Utilisation de l'Algèbre (Max, +) et des réseaux de Petri Z/pZ dans l'étude d'un processus industriel. Thèse de doctorat, Laboratoire d'Ingénierie des Systèmes Automatisés, Université d'Angers, 2002.

Mabed, M. ; Bousseau, F. and M. Bourcerie. Les RdP Z/pZ un outil d'apprentissage et d'intégration des perturbations. 4ème Conférence Francophone de Modélisation et de Simulation. "Organisation et conduite d'activités dans l'industrie et les services MOSIM03. Avril, 2003, Toulouse, France

Manier, M-A. ; and P. Baptiste. Etat de l'art : Ordonnancement de robots de manutention en galvanoplastie. AIPI RAIRO, vol. 28.n°1. pp. 7-35, 1994.

Marteau, S.; and M. Bourcerie, Complex sequencing by methods of interpolation in Z/pZ colored Petri nets, 4th International Conference on Emerging Technologies and Factory Automation INRIA / IEEE ETFA'95, p 573-581, France, 1995.

Marteau, S.; and M. Bourcerie, Dynamic evolution of Z/pZ colored Petri nets models. In : Computational Engineering in Systems Applications, Symposium on Discrete Events and Manufacturing Systems (IEEE-SMC CESA'96), p. 48-52, Lille, France, 1996.

Mares, M. Sur la modélisation et l'analyse des systèmes complexes à événements discrets par réseaux de Petri. Thèse de doctorat, Laboratoire d'Ingénierie des Systèmes Automatisés, Institut des Sciences et Techniques de l'Ingénieur d'Angers, 1997.

Morel, J-Y. Contribution à la modélisation et à l'analyse des systèmes à événements discrets par réseaux de Petri généralisés et colorés. Thèse de Doctorat, Laboratoire d'Ingénierie des Systèmes Automatisés – Institut des Sciences et Techniques de l'Ingénieur d'Angers –, 1996.

Morel, J-Y.; M. Mares and M. Bourcerie, Composition and analysis of generalised and colored Petri nets. Multiconference IEEE IMACS, Computational Engineering in Systems Applications, CESA' 96, France, 1996.

Proth, J-M ; and X. Xie, Les réseaux de Petri pour la conception et la gestion de systèmes de production, Masson, Paris, 1994.

Zimmermann, A. A modeling method for flexible manufacturing systems based on colored Petri nets. In proc. Int Workshop on new directions of control and manufacturing, p. 147-154, Hong Kong, 1994.

SIMULATING QUANTUM INTERFERENCE IN FEYNMAN'S $\sqrt{\text{NOT}}$-COMPUTER WITH STOCHASTIC PETRI NETS

Leo Ojala
Olli-Matti Penttinen
Helsinki University of Technology
Laboratory for Theoretical Computer Science
P.O. Box 9700, FIN-02015 TKK, Finland
Email: Leo.Ojala@hut.fi
Email: Olli-Matti.Penttinen@luukku.com

> It is as if we had invented a machine that first scrambles eggs and then unscrambles them.
>
> Brian Hayes (1995)

ABSTRACT

It is possible to consider quantum interference as a new kind of useful resource that quantum mechanics brings out to computing. In this paper, we demonstrate this by simulating the free quantum evolution of simple quantum circuits consisting of $\sqrt{\text{NOT}}$-gates. The simulation is carried out with a novel method called Stochastic Differential Analysis. It is based on using Stochastic Petri Nets to simulate the solution of ordinary differential equations.

INTRODUCTION

In quantum computing, we have a cross-disciplinary merge of the most significant developments in science and technology of the twentieth century: *quantum mechanics* and *computing*. Quantum computation is based on two quantum phenomena: *quantum interference* and *quantum entanglement* (Deutsch et al. 2000; Gruska 1999). In this paper, we demonstrate the origin and behavior of the first phenomenon in detail. Our object of demonstration is a simple quantum gate $\sqrt{\text{NOT}}$. It's behavior, free evolution, is simulated using a stochastic differential analyzer (SDA). When two $\sqrt{\text{NOT}}$-gates are connected in series and free evolution of the resulting NOT-circuit simulated, the quantum interference phenomenon can clearly be observed and explained.

The novel simulation method based on the use of SDA has been used in our companion paper (Ojala et al. 2003) to simulate a quantum swap-circuit and theoretically explained in (Penttinen 2003).

THE QUANTUM $\sqrt{\text{NOT}}$-GATE

A *quantum system* consists of one or more quantum bits, *qubits*. The state of the qubit is written as

$$\alpha \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \alpha, \beta \in \mathbb{C} : \ |\alpha|^2 + |\beta|^2 = 1 \ .$$

As it can be seen, a qubit can be in a superposition of the *eigenstates* $\left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$. The complex coefficients α and β are called *probability amplitudes*. Also, according to quantum mechanics, if the state of qubit were measured, the result $\left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]$ is achieved with a probability $|\alpha|^2$ and the result $\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$ with probability $|\beta|^2$.

The $\sqrt{\text{NOT}}$-gate operates on one input qubit and has the property two consecutive applications of $\sqrt{\text{NOT}}$-gates produces the same result as the classical NOT-gate, a transform of the state $\left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]$ to $\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$ and the state $\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$ to $\left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]$. The matrix representation of $\sqrt{\text{NOT}}$ is

$$A = \begin{bmatrix} \lambda & \omega \\ \omega & \lambda \end{bmatrix} \ , \quad \lambda = \frac{1+i}{2} \ , \quad \omega = \frac{1-i}{2} \ . \tag{1}$$

The effect of applying $\sqrt{\text{NOT}}$ to the basis states $\left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$ is

$$\begin{aligned} \sqrt{\text{NOT}} \begin{bmatrix} 1 \\ 0 \end{bmatrix} &= \begin{bmatrix} \lambda & \omega \\ \omega & \lambda \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \lambda \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \omega \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda \\ \omega \end{bmatrix} , \\ \sqrt{\text{NOT}} \begin{bmatrix} 0 \\ 1 \end{bmatrix} &= \begin{bmatrix} \lambda & \omega \\ \omega & \lambda \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \omega \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \lambda \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \omega \\ \lambda \end{bmatrix} . \end{aligned} \tag{2}$$

The state of the qubit after just one $\sqrt{\text{NOT}}$-gate operation is the superposition of the states $\left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$. Since $|\lambda|^2 = |\omega|^2 = 1/2$, it appears that a measurement of the state of the qubit gives $\left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]$ and $\left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$ randomly with equal probability quite independently of the value of the input (Gruska 1999; Williams and Clearwater 1998). But, when two $\sqrt{\text{NOT}}$-gates are connected in series, the output value is *always* the negation of the input; a perfect NOT-operation is performed.

How is this paradoxical behavior possible? It is our aim in this paper to study both behaviors by simulation and try to find an explanation for this phenomenon.

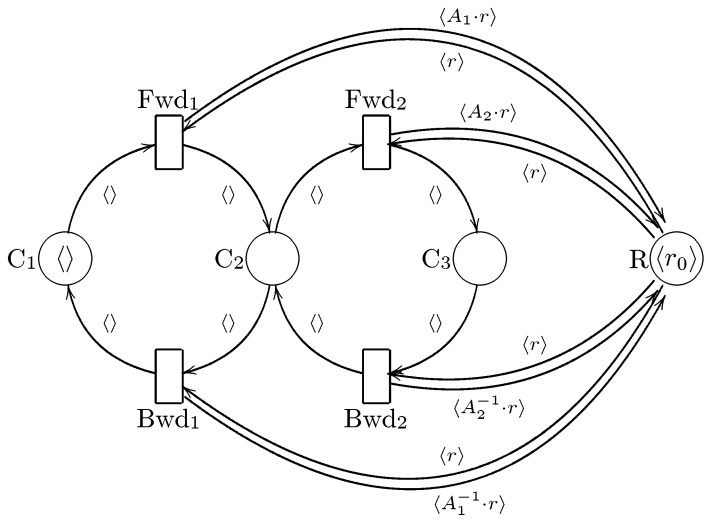


Figure 1: A classical reversible serial computer architecture modeled using Pr/T-nets. Also the initial marking is given.

REVERSIBILITY

One of the basic properties of quantum computing is the *reversibility of operations* (Gruska 1999; Williams and Clearwater 1998). A logic gate is reversible if and only if the knowledge of the output is sufficient to reconstruct the state of the input; i.e., no amount of information is lost when the gate is applied to the input. For example, the classical NOT-gate is reversible, whereas all classical binary gates such as the AND-, OR- and XOR-gate are irreversible.

In quantum mechanics, the operations are usually given in matrix form (cf. (1)); in linear algebraic terms, the requirement of reversibility implies that the matrix must be *unitary*.

In Fig. 1, we give an example of a simple classical reversible serial computer architecture that is modeled using Pr/T-nets. The net in Fig. 1 is a model of applying the reversible operations A_1 and A_2 in succession to a variable r, whose initial value is r_0, The value of r is held in an input–output memory register throughout the computation. There are actually two coordinating sub-nets glued together by their transitions: the "cursor net" around the places C_1, C_2, and C_3 ensures the correct ordering of gate applications, and the "computation net" with place R carries out the actual computations. Transitions Fwd_1 and Fwd_2 model computation in the forward direction, and transitions Bwd_1 and Bwd_2 in the reverse direction.

The cursor net has a place invariant: in any reachable marking, exactly one of the places C_1, C_2, and C_3 is marked with an empty tuple $\langle\rangle$, the other two being unmarked. Whenever place C_3 is marked, place R holds the final value of r and the computation ought to be stopped.

The net can also be considered as a specification of the *logical behavior* of a two-staged quantum circuit. If we let the $\sqrt{\text{NOT}}$-operation in (1) correspond to operations A_1 and A_2 and a single qubit to R in Fig. 1, we have a logical description of the reversible $\sqrt{\text{NOT}} \cdot \sqrt{\text{NOT}}$-computation. Actually, we have given a model of the logical behavior of a two-gate *Feynman's quantum computer* (FQC) (Feynman 1985), which can be used as a platform for our simulation task.

FEYNMAN'S QUANTUM COMPUTER

In addition to M input–output qubits, a N-gate FQC has $N+1$ cursor qubits (cf. Fig. 1), resulting in a state space of $2^{(N+1+M)}$ eigenstates. Because of the cursor invariant, only $K = (N+1) \cdot 2^M$ states are reachable. The reachable states of $\sqrt{\text{NOT}}$ and $\sqrt{\text{NOT}} \cdot \sqrt{\text{NOT}}$ FQCs are given in Table 1.

The free evolution of a FQC obeys the *Schrödinger differential equation*

$$\frac{\mathrm{d}}{\mathrm{d}t}|\psi(t)\rangle = -\frac{\mathrm{i}}{\hbar}\mathsf{H}|\psi(t)\rangle \qquad (\hbar = h/2\pi) \ ,$$

where h is the Planck constant and H is a Hermitian matrix called *the Hamiltonian*. $|\psi(t)\rangle$ denotes the quantum state of the system, which is a superposition

$$|\psi(t)\rangle = \sum_{i=1}^{K} \alpha_i |\psi_i(t)\rangle \ , \qquad \alpha_i \in \mathbb{C}$$

Table 1: The reachable eigenstates of $\sqrt{\text{NOT}}$ and $\sqrt{\text{NOT}} \cdot \sqrt{\text{NOT}}$ FQCs

$\sqrt{\text{NOT}}$ FQC			
name	*vector*	*cursor*	*data*
$\|\psi_1\rangle$	$[1\ 0\ 0\ 0]^\mathsf{T}$	initial	**0**
$\|\psi_2\rangle$	$[0\ 1\ 0\ 0]^\mathsf{T}$	initial	**1**
$\|\psi_3\rangle$	$[0\ 0\ 1\ 0]^\mathsf{T}$	final	**0**
$\|\psi_4\rangle$	$[0\ 0\ 0\ 1]^\mathsf{T}$	final	**1**

$\sqrt{\text{NOT}} \cdot \sqrt{\text{NOT}}$ FQC			
name	*vector*	*cursor*	*data*
$\|\psi_1\rangle$	$[1\ 0\ 0\ 0\ 0\ 0]^\mathsf{T}$	initial	**0**
$\|\psi_2\rangle$	$[0\ 1\ 0\ 0\ 0\ 0]^\mathsf{T}$	initial	**1**
$\|\psi_3\rangle$	$[0\ 0\ 1\ 0\ 0\ 0]^\mathsf{T}$	intermediate	**0**
$\|\psi_4\rangle$	$[0\ 0\ 0\ 1\ 0\ 0]^\mathsf{T}$	intermediate	**1**
$\|\psi_5\rangle$	$[0\ 0\ 0\ 0\ 1\ 0]^\mathsf{T}$	final	**0**
$\|\psi_6\rangle$	$[0\ 0\ 0\ 0\ 0\ 1]^\mathsf{T}$	final	**1**

with the probability amplitudes α_i having the normalizing condition

$$\sum_{i=1}^{K} |\alpha_i|^2 = 1 \ .$$

The labeling of states and the operator matrices determine the Hamiltonian up to phase angles of probability amplitudes. For simulation, we add the additional requirement that all phase angles have to be the same as those obtained by applying the gate operators discretely (cf. (2)). With $A_1 = A_2 = A$ in (1), the Hamiltonians are

$$H_{\sqrt{\text{NOT}}} = \begin{bmatrix} 0 & (\mathrm{i}A_1)^\dagger \\ \mathrm{i}A_1 & 0 \end{bmatrix} = \left[\begin{array}{cc|cc} 0 & 0 & -\lambda & \omega \\ 0 & 0 & \omega & -\lambda \\ \hline -\omega & \lambda & 0 & 0 \\ \lambda & -\omega & 0 & 0 \end{array}\right]$$

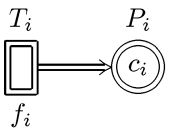
and

$$H_{\sqrt{\text{NOT}}\cdot\sqrt{\text{NOT}}} = \begin{bmatrix} 0 & (\mathrm{i}A_1)^\dagger & 0 \\ \mathrm{i}A_1 & 0 & (\mathrm{i}A_2)^\dagger \\ 0 & \mathrm{i}A_2 & 0 \end{bmatrix}$$

$$= \left[\begin{array}{cc|cc|cc} 0 & 0 & -\lambda & \omega & 0 & 0 \\ 0 & 0 & \omega & -\lambda & 0 & 0 \\ \hline -\omega & \lambda & 0 & 0 & -\lambda & \omega \\ \lambda & -\omega & 0 & 0 & \omega & -\lambda \\ \hline 0 & 0 & -\omega & \lambda & 0 & 0 \\ 0 & 0 & \lambda & -\omega & 0 & 0 \end{array}\right] .$$

STOCHASTIC DIFFERENTIAL ANALYSIS

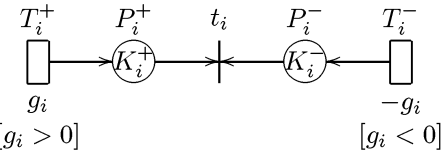
In SDA, solving a system of linear ordinary differential equations (ODE) is simulated with Generalized Stochastic Petri Nets (GSPN) (Ojala et al. 2003; Penttinen 2003; Ajmone Marsan et al. 1995). Each equation $y_i' = a_{i1}y_1 + \cdots + a_{iN}y_N + b_i$ is modeled with a net fragment



(3)

$$f_i = a_{i1}\#P_1 + \cdots + a_{iN}\#P_N + b_i$$

where each P_i models the value of y_i and c_i models the initial value of y_i. The SDA net (3) has an underlying GSPN representation



$$g_i = a_{i1}(\#P_1^+ - \#P_1^-) + \cdots + a_{iN}(\#P_N^+ - \#P_N^-) + b_i$$

where g_i and $-g_i$ denote the marking dependent firing rates of transitions T_i^+ and T_i^-. The expression $\#P$ denotes the number tokens in place P in a marking. The expressions $[g_i \gtrless 0]$ act as *transition guards* that must evaluate to true for a transition to have concession in a marking. Assuming that $c_i \in \mathbb{Z}$, the initial markings are such that $c_i = K_i^+ - K_i^-$. For non-integral initial values (not needed in our simulations), a suitable *initial distribution* is set up with the help of an auxiliary place and two conflicting immediate transitions.

SIMULATION

We are going to simulate the operations of both FQCs with input **0**, i.e., with $|\psi_1\rangle$ as the initial state. Since our simulator cannot work with complex numbers, we have to split the Schrödinger equations into their real and imaginary parts. To further simplify the equations, we also scale time by $\hbar$ by letting $\mathbf{z}(t) = |\psi(\hbar t)\rangle$ and $z_i = x_i + \mathrm{i}y_i$. Now, the Schrödinger equations for the $\sqrt{\text{NOT}}$ computer are

$$\begin{aligned}
x_1' &= \tfrac{1}{2}(-x_3 - y_3 - x_4 + y_4) \\
y_1' &= \tfrac{1}{2}(\ \ x_3 - y_3 - x_4 - y_4) \\
x_2' &= \tfrac{1}{2}(-x_3 + y_3 - x_4 - y_4) \\
y_2' &= \tfrac{1}{2}(-x_3 - y_3 + x_4 - y_4) \\
x_3' &= \tfrac{1}{2}(\ \ x_1 - y_1 + x_2 + y_2) \\
y_3' &= \tfrac{1}{2}(\ \ x_1 + y_1 - x_2 + y_2)
\end{aligned}$$

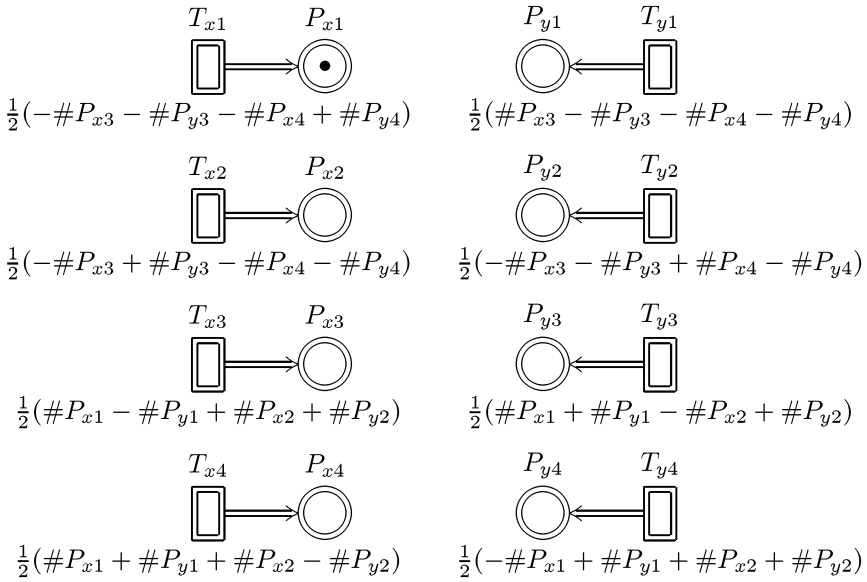
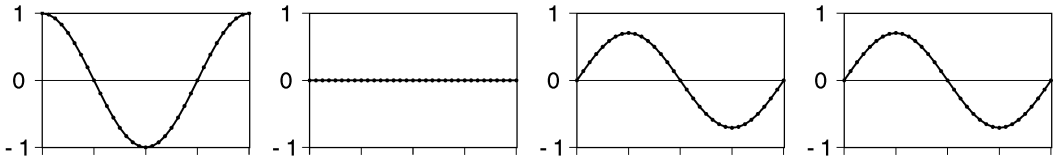


Figure 2: A SDA net of the $\sqrt{\text{NOT}}$ FQC with input 0



(a) z_1 (b) z_2 (c) $z_3/\lambda\sqrt{2}$ (d) $z_4/\omega\sqrt{2}$

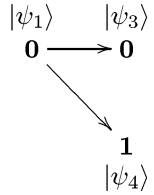
Figure 3: State probability amplitudes of the $\sqrt{\text{NOT}}$ FQC vs. time for $t \in [0, 2\pi]$. Intervals between tick marks are $\pi/2$

$$x_4' = \tfrac{1}{2}(\ \ x_1 + y_1 + x_2 - y_2)$$
$$y_4' = \tfrac{1}{2}(-x_1 + y_1 + x_2 + y_2)$$

A SDA net of the $\sqrt{\text{NOT}}$ FQC is given in Fig. 2 and the results of a simulation up to $t = 2\pi$ in Fig. 3. The simulation results were obtained by performing a transient analysis of the GSPN underlying the SDA net with our experimental SDA software, which is capable of computing average place markings of an unbounded GSPN.

All the curves in Fig. 3 have a form that is to be expected. z_1, the amplitude of $|\psi_1\rangle$ varies between -1 and 1. Except for the isolated points $t = \pi/2 + n \cdot \pi$, there is a positive probability that upon measurement the FQC is found in its initial state. At $t = n \cdot \pi$, the probability is 1. z_2 is identically 0 as it should when the computer is started in state $|\psi_1\rangle$. To be able to graph them against a real axis, the amplitudes z_3 and z_4 have been divided by unit complex numbers $\lambda\sqrt{2} = (1+\mathrm{i})/\sqrt{2}$ and $\omega\sqrt{2} = (1-\mathrm{i})/\sqrt{2}$, respectively. The absolute values of z_3 and z_4 are always the same, consistent with the fact that the $\sqrt{\text{NOT}}$-gate should produce outcomes **0** and **1** with equal probability.

From an operational point of view, the curves may be interpreted as follows: starting with input **0**, there are two separate forward computations that the $\sqrt{\text{NOT}}$-computer may perform.



Path $\mathbf{0} \to \mathbf{0}$ leads to state $|\psi_3\rangle$, and path $\mathbf{0} \to \mathbf{1}$ to state $|\psi_4\rangle$. However, we must not err to believe that the FQC chooses one of these paths at random. There is nothing random about the Schrödinger differential equation that governs the free evolution of a quantum system. Instead, we have to accept that *both paths are chosen* in parallel. This evolution leads to a superposition of states which upon *measurement* collapses to either one of the eigenstates with equal probability.

With the same variable substitutions as above, the Hamiltonian of the $\sqrt{\text{NOT}} \cdot \sqrt{\text{NOT}}$-computer leads to an ODE-system

$$x_1' = \tfrac{1}{2}(-x_3 - y_3 - x_4 + y_4)$$
$$y_1' = \tfrac{1}{2}(\ \ x_3 - y_3 - x_4 - y_4)$$
$$x_2' = \tfrac{1}{2}(-x_3 + y_3 - x_4 - y_4)$$

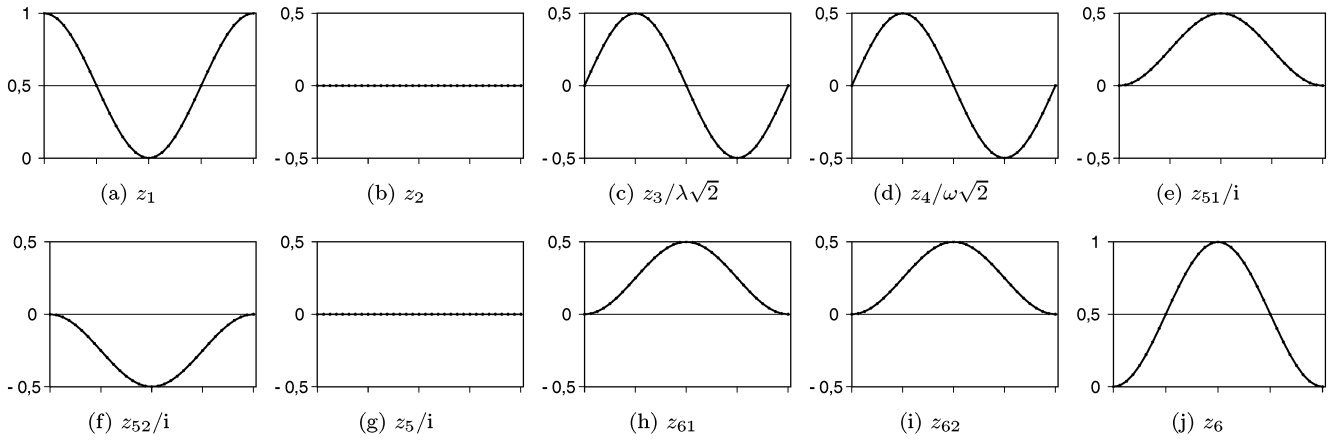
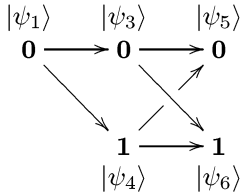


Figure 4: State probability amplitudes of the $\sqrt{\text{NOT}} \cdot \sqrt{\text{NOT}}$ FQC vs. time for $t \in \left[0, \pi\sqrt{2}\right]$. Intervals between tick marks are $\pi/2\sqrt{2}$

$$
\begin{aligned}
y'_2 &= \tfrac{1}{2}(-x_3 - y_3 + x_4 - y_4) \\
x'_3 &= \tfrac{1}{2}(\ \ x_1 - y_1 + x_2 + y_2 - x_5 - y_5 - x_6 + y_6) \\
y'_3 &= \tfrac{1}{2}(\ \ x_1 + y_1 - x_2 + y_2 + x_5 - y_5 - x_6 - y_6) \\
x'_4 &= \tfrac{1}{2}(\ \ x_1 + y_1 + x_2 - y_2 - x_5 + y_5 - x_6 - y_6) \\
y'_4 &= \tfrac{1}{2}(-x_1 + y_1 + x_2 + y_2 - x_5 - y_5 + x_6 - y_6) \\
x'_5 &= \tfrac{1}{2}(\ \ x_3 - y_3 + x_4 + y_4) \\
y'_5 &= \tfrac{1}{2}(\ \ x_3 + y_3 - x_4 + y_4) \\
x'_6 &= \tfrac{1}{2}(\ \ x_3 + y_3 + x_4 - y_4) \\
y'_6 &= \tfrac{1}{2}(-x_3 + y_3 + x_4 + y_4)
\end{aligned}
$$

Before proceeding to simulate this, we construct the potential forward computation paths of the computer.



To be able to separate the two paths $\mathbf{0} \to \mathbf{0} \to \mathbf{0}$ and $\mathbf{0} \to \mathbf{1} \to \mathbf{0}$ that lead from $|\psi_1\rangle$ to $|\psi_5\rangle$ from each other and the paths $\mathbf{0} \to \mathbf{0} \to \mathbf{1}$ and $\mathbf{0} \to \mathbf{1} \to \mathbf{1}$ that lead from $|\psi_1\rangle$ to $|\psi_6\rangle$ from each other, we split the equations of amplitudes z_5 and z_6 into two subparts.

$$
\begin{aligned}
x'_{51} &= \tfrac{1}{2}(\ \ x_3 - y_3) & y'_{51} &= \tfrac{1}{2}(\ \ x_3 + y_3) \\
x'_{52} &= \tfrac{1}{2}(\ \ x_4 + y_4) & y'_{52} &= \tfrac{1}{2}(-x_4 + y_4) \\
x_5 &= x_{51} + x_{52} & y_5 &= y_{51} + y_{52} \\
x'_{61} &= \tfrac{1}{2}(\ \ x_3 + y_3) & y'_{61} &= \tfrac{1}{2}(-x_3 + y_3) \\
x'_{62} &= \tfrac{1}{2}(\ \ x_4 - y_4) & y'_{62} &= \tfrac{1}{2}(\ \ x_4 + y_4) \\
x_6 &= x_{61} + x_{62} & y_6 &= y_{61} + y_{62}
\end{aligned}
$$

Here, z_{51} stands for the computation that leads from $|\psi_3\rangle$ to $|\psi_5\rangle$, z_{52} from $|\psi_4\rangle$ to $|\psi_5\rangle$, etc. In the interest of brevity, we omit the construction of an SDA-representation of the $\sqrt{\text{NOT}} \cdot \sqrt{\text{NOT}}$-computer from the paper. Instead, we give the curves that are the outcome of a simulation of the net up to $t = \pi\sqrt{2}$. The results are seen in Fig. 4.

Although the form (and the interpretation) of amplitudes z_1 to z_4 is the same as in Fig. 3, the actual values are quite different from each other. This is due to the fact that in the $\sqrt{\text{NOT}}$-computer, computation can proceed only backwards from states $|\psi_3\rangle$ and $|\psi_4\rangle$, whereas in the $\sqrt{\text{NOT}} \cdot \sqrt{\text{NOT}}$-computer, it proceeds in both directions.

More importantly, the interference patterns concerning final states $|\psi_5\rangle$ and $|\psi_6\rangle$ are clearly visible in the figure. The correct operation of the computer requires that the amplitude of state $|\psi_5\rangle$ has to be identically zero, when the computer is started in state $|\psi_1\rangle$. As the reader can see, the sum of amplitudes z_{51} and z_{52} representing computation paths $\mathbf{0} \to \mathbf{0} \to \mathbf{0}$ and $\mathbf{0} \to \mathbf{1} \to \mathbf{0}$, respectively, is indeed zero. This phenomenon is an example of *destructive interference*. On the other hand, amplitudes z_{61} and z_{62} representing computations $\mathbf{0} \to \mathbf{0} \to \mathbf{1}$ and $\mathbf{0} \to \mathbf{1} \to \mathbf{1}$, are always equal; and, therefore, amplify each other. This is an example of *constructive interference*. Hopefully, this also given a plausible explanation to why it is vitally important to carry out the simulations with complex amplitudes, although only their absolute values play a role in determining state probabilities.

CONCLUSIONS

In this paper, we have demonstrated by simulation the effects of constructive and destructive interference on the computations a simple quantum gate performs. For the simulations, we have used a novel method, SDA, for solving ODE-systems with Stochastic Petri Nets, In the

future, we will extend our approach by applying our approach to more complex quantum computing architectures, such as *quantum cellular automata* (Margolus 1990).

REFERENCES

Ajmone Marsan, O.; G. Balbo; G. Conte; S. Donatelli and G. Francheschinis. 1995. *Modelling with Generalized Stochastic Petri Nets.* John Wiley & Sons, New York.

Deutsch, D. 1985. "Quantum Theory, the Church–Turing Principle and the Universal Quantum Computer". *Proceedings of the Royal Society London A*, 400, 97-117.

Deutsch, D.; A. Ekert and R. Lupacchini. 2000. "Machines, Logic and Quantum Physics". *Bulletin of Symbolic Logic*, 6(3), September 2000, 265–283.

Feynman, R. 1985. "Quantum Mechanical Computers". *Optic News*, February 1985, 11–20. Reprinted in *Foundations of Physics*, 16(6), 1986, 507–531.

Gruska, J. 1999. *Quantum Computing.* McGraw–Hill, UK.

Margolus, N. 1990. "Parallel Quantum Computation". In *Complexity, Entropy, and the Physics of Information*, vol. VIII of *SFI Studies in the Sciences of Complexity*, W.H. Zurek (Ed.). Addison-Wesley, 273–287.

Ojala, L.; O. Penttinen and H. Rantanen. 2003. "A Novel Application of Petri Nets: Simulation of Serial Quantum Computers—Feynman's Swap Computer". In *Proceedings of ICALP03 Satellite Workshop on Stochastic Petri Nets* (Eindhoven, the Netherlands). Universität Dortmund, Fachbereit Informatik, Forschungsbericht Nr. 780, 103–122.

Penttinen, O. 2003. "On Solving Ordinary Differential Equation Systems with Generalized Stochastic Petri Nets". In *Proceedings of ESMC03* (Naples, Italy).

Williams, C. and S. Clearwater. 1998. *Explorations in Quantum Computing.* Springer-Verlag.

PETRI NETS
II

Petri Net distributed simulation using HLA based on Petri net components

Stéphanie Combettes, Alexandre Nketsa
LAAS-CNRS
7, avenue du colonel Roche, 31077 Toulouse cedex 4, France
e-mail: {scombett, alex}@laas.fr

Keywords
Component based Development (CBD), HLA, Petri Net

ABSTRACT
This paper presents an approach to build components based on Petri Nets for distributed simulation using HLA. The components built during the design stage are strongly based on the reuse of preexisting components. The component approach allows to exploit the inherent hierarchy of Petri Nets by encapsulating lower levels. The first part of this paper deals with the rules and the methodology used to build components. The second part introduces an example with an HLA implementation.

INTRODUCTION
Simulation of complex systems by means of computerised models is shifting to new paradigms like Component Based Development (CBD) and Distributed Component Computing (DCC) where software components can be used to assemble simulations from a variety of heterogeneous services and models.
CBD refers to the techniques and tools [Cenzis and Oguztuzun, 2002, Radeski et al., 2002] that allow the construction of an application from both new and preexisting components. HLA (*High Level Architecture*) can be seen as a component architecture. An HLA federation is a component-based application where federates represent the components. Consequently, CBD process is applied to federate level. A federate can be built from reusable components. These components can be connected to each other at run time to form a federate. The use of CBD approach demands:

- a cautious selection of components that should be used in a specific situation to assure a minimal integration and parametrisation effort,
- a correct component parametrisation to commit it to the final context,
- a careful interconnection with the other components to obtain the desired systems.

Petri Nets offer some properties to overcome these demands. Distributed simulation of Petri Nets has been considered for years as an attractive idea for increasing the effectiveness of complex model execution. New technologies put emphasis no longer on speed up, but rather on the possibility of constructing and executing complex models from heterogeneous components developed and run on different machines.

One of the problems arising from the CBD approach is related to the specification of a parametric and generic component capable of being instantiated within a set of well-characterised behavioural class. To solve this problem, we propose to split the component in many layers.
In this paper, an approach to build Petri-Net components and their integration in an HLA distributed simulation is proposed. The rest of the paper is organised as follows: firstly, we expose the construction of a Petri Net component and give its formal definition. Secondly, we deal with the coupling of the components and we enumerate the coupling rules and the closure conditions as well as defining the Petri Net Component structure. Thirdly we introduce an example with the HLA implementation.

BUILDING PETRI NET COMPONENT
A Petri Net Component represents a partial or complete system view. It also represents a system logical view and enables to simulate it. It is made of a set of components, each of them corresponding to a sub-system. These sub-systems are hierarchically organised by abstraction levels and they lead to implementation of modularity. The component encapsulates sub-systems by incorporating all the communication primitives required for distributed simulation and by offering an interoperability definition frame. Moreover this component integrates the supports needed for distributed simulation implementation. Now our formal definition of a Petri Net component is exposed.

Formal Definition
A Petri Net Component may be made of sub-systems called components. It is either an atomic Petri Net Component or a coupled one. A coupled Petri Net Component may consist of a set of atomic and/or coupled Petri Net Component linked together. Components coupling forms a new component. An atomic Petri Net Component then a coupled one are exposed.

Atomic Petri Net Component
An atomic Petri Net Component is the minimum element that a component can be. Here is its definition :

$$PNC_{atomic} = < X_{ext}, Y_{ext}, X, Y, PN, \delta_{translExt-Int}, \delta_{translInt-Ext}, C_{interop}, C_{communication} > \quad (1)$$

with
X_{ext} : set of external input events ;
Y_{ext} : set of external output events ;

X : set of internal input events ;
Y : set of internal output events ;
PN : internal part of atomic PNC where :
$PN =< P, T, A, M >$ with :
P : finite set of places ;
T : finite set of transitions ;
$A : T \times P \cup P \times T$: set of arcs linking places to transitions and vice versa ;
M : initial marking ;
$\delta_{translExt-Int} : X_{ext} \longrightarrow X$. Translation function translating external variables into internal variables ;
$\delta_{translInt-Ext} : Y \longrightarrow Y_{ext}$. Translation function translating internal variables into external variables ;
$C_{communication}$: set of communications made of three parts : P_{com}^{IorO}, sets of communication ports, and Δ_{com} set of communication primitives ;
$C_{interop}$: interoperability set including messages and links existing between them. This set consists of:
$Cinter - Type(m)$: it returns type of event m.
$Cinter - Link(m)$: it returns links that event m has with events of different type.
$C_{Structure}$: a global structure specifying data characteristics.
Event m may belong indifferently to X_{ext}, Y_{ext}, X or Y.

Coupled Petri Net Component

Our definition of a coupled Petri Net Component (PNC) is similar to the atomic component one. Moreover it takes into account the existing coupling between sub-components. A coupled component is :

$$PNC_{coup} = < X_{cext}, Y_{cext}, E_c, C, X_c, Y_c, \delta_{link}, \delta_{translExt-Int}, \delta_{translInt-Ext}, C_{interop}, C_{communication} > \quad (2)$$

with :
X_{cext} : set of external input events ;
Y_{cext} : set of external output events ;
E_c : set of internal states of PNC_{coup} ;
C : set of model components with :
X_c^i : set of component external inputs C^i,
Y_c^i : set of component external outputs C^i,
X_c : set of internal input events of PNC_{coup} ;
Y_c : set of internal output events of PNC_{coup}.
$\delta_{link} : X_c^i \longrightarrow Y_c^j$ or $Y_c^i \longrightarrow X_c^j$ or
$\varnothing$ if no link , with $i \neq j$.
Unspecified elements are identical to the ones previously defined. We will specify the members of X_c and Y_c when we deal with coupling in the next section.
We notice that the coupled PNC is made of three parts. The first one represents the internal part (E_c, C, X_c, Y_c, δ_{link}), the second one concerns fitting and translation ($\delta_{translExt-Int}$, $\delta_{translInt-Ext}$, $C_{interop}$) and the third one is called the external part (X_{cext}, Y_{cext}, $C_{communication}$). This point will be developed in the subsection dealing with the structure of PNC. Now we expose coupling rules.

COUPLING

A Petri Net component may be made of sub-systems (atomic or coupled components) that are linked together. Component coupling forms a new component. Generally, defining this new component with the same formalism is a huge advantage [Zeigler et al., 1999]. We demonstrate the closure under coupling condition to prove that the coupling of our defined components yields to a new component.

Coupling rules

We assume that in a coupled component, communication between subnets is asynchronous and based on communication places. PNCs contain internal data that don't take place in coupling. Only the external (input and output) data and the communication ports are considered in coupling.
Coupling is a set of links created between external events. The events are required to be homogeneous (identical type and format) and the link is done between communication ports. In fact coupling comprises two steps.
Firstly *structural coupling* takes place in linking a PNC_i input port with one or more PNC_j output ports (ports connections). Thus PNC_i external data are linked to PNC_j internal data. As a consequence data must have the same nature and same type [Combettes, 2003].
Secondly *semantical coupling* guarantees the consistency of exchanged messages (interoperability). As PNC structure contains a translation function, data fitting is possible.
The two coupling conditions are defined:

- **Structural coupling condition** : An *output* port must be only linked to one or several *input* ports. Two ports having the same type (input ports or output ports) cannot be linked together.
- **Semantical coupling condition** :

$$\forall\, x^i \in X^i\ , X^i \in C^i \text{ and } \forall\, y^j \in Y^j\ , Y^j \in C^j$$
$$\text{then, } C^i_{inter-Type}(x^i) = C^j_{inter-Type}(y^j) \quad (3)$$

The satisfaction of the two previous conditions is necessary but not sufficient for the feasibility of the communication between PNCs.
PNC may be coupled in several ways. Coupled components result in a new system. We wish this new system to be a new PNC. In this way different couplings for PNCs can be exposed and input sets can be deduced. Reasoning is similar for the output set. The following study was inspired by a demonstration made for Petri Nets [Kiehn, 1989]. Different ways of coupling PNCs are showed. Demonstration of these couplings are given in [Combettes and Nketsa, 2002].

Definition 1 : **Concatenation**
Two Petri Net Components (PNC1 and PNC2) are in sequence.

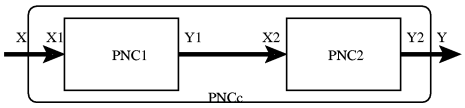


Figure 1: coupling under concatenation

Definition 2 : **Shuffle**
Two Petri Net Components (PNC1 and PNC2) are in parallel.

Definition 3 : **Union**
Two Petri Net Components (PNC1 and PNC2) are put in union.

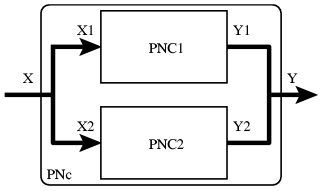


Figure 2: coupling under shuffle

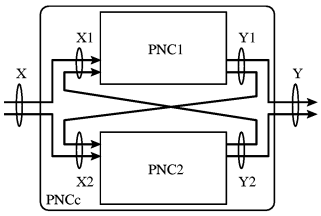


Figure 3: coupling under union

Definition 4 : **Intersection**
Two Petri Net Components (PNC1 and PNC2) are in intersection.

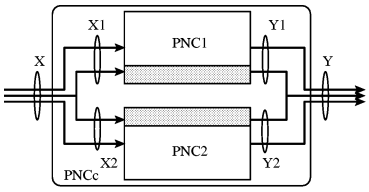


Figure 4: coupling under intersection

In the above definitions, while exposing the different manners of PNC coupling, the internal states must be explicited according to the coupled component internal states. We consider that the internal states of the resulting system is the union of internal states of each encapsulated PNC. From which the mathematical formula is :
for n PNCs,

$$E_c = \bigcup E_c^i \qquad \text{where } i \in [0; n-1] \qquad (4)$$
$$\text{with } E_c^i \text{ set of internal states of } PNC^i.$$

We underline the fact that coupling between PNCs may be a composition of different couplings explained before. In the next subsection we deal with closure conditions.

Closure conditions

By defining an atomic and a coupled PNC identically, we can obtain coupled PNCs that respect closure under coupling conditions. The closure properties allow to consider coupled PNC as a new component and to describe it with the same formalism as an atomic one. As a consequence coupled PNC may be used again in a new coupling [Zeigler et al., 1999] [Kiehn, 1989]. Closure conditions are demonstrated with the use of previous definitions.
Closure under concatenation condition: a coupled PNC encapsulates PNCs in sequence: outputs of a PNC are linked to inputs of the next PNC. We deduce by applying definition 1 that : for n PNCs,

$$X_c = X^1 \text{ and } Y_c = Y^n,$$
$$\delta_{link}(X^i) = Y^{i+1} \ , \ \delta_{link}(Y^{i+1}) = X^i,$$
$$\text{with } (X^i, Y^i) \in (C^i, C^i)\ (C^i, C^i) \in C,\ i \in [0; n-1].$$

Closure under shuffle condition: coupled PNC encapsulates parallel PNCs. Sub-PNS are not really linked. We deduce by using definition 2 that :

$$X_c = \bigcup X^i \text{ and } Y_c = \bigcup Y^i,$$
$$\delta_{link}(x) = \varnothing \text{ with } x \in X^i,$$
$$\delta_{link}(y) = \varnothing \text{ with } y \in Y^i,$$
$$(X^i, Y^i) \in C^i,\ \forall\ C^i \in C,\ i \in [0; n-1].$$

Closure under union condition: a coupled PNC encapsulates PNCs coupled together : some outputs of a PNC are connected to some inputs of other ones. We deduce by using definition 3 that :

$$X_c = \bigcup X^i - (X^i \cap Y^j)\ ,\ Y_c = \bigcup Y^i - (Y^i \cap X^j),$$
$$\delta_{link}(x^i) = y^j \text{ and } \delta_{link}(y^i) = x^j$$
$$\text{with } (x^i, y^j) \in (X^i, Y^j) \in (C^i, C^j)\ \forall\ (C^i, C^j) \in C,$$
$$(i, j) \in [0; n-1], i \neq j.$$

Closure under intersection condition:
a coupled PNC encapsulates PNCs having at least one identical data as input and/or output. By applying definition 4 we deduce that :

$$X_c = \bigcup X^i \text{ and } Y_c = \bigcup Y^i$$
$$\delta_{link}(x) = \varnothing \text{ and } \delta_{link}(y) = \varnothing$$
$$\text{with } (x, y) \in (X^i, Y^i) \in C^i,\ \forall\ C^i \in C,$$
$$i \in [0; n-1].$$

A resulting coupled PNC respecting these (one or more) closure conditions may be used to be coupled, since it respects closure under coupling condition.

Structure of The Petri Net Component

The PNC approach allows to define a separation between the implementation part and the layers needed for distributed simulation. Consequently, the model to be simulated may be validated independent from the architecture. In view of its general characteristics combined to its formal definition, a PNC may be organised according to three layers.
The first one is the **Linkage layer**. It ensures synchronisation and exchanges with other sites or between sub-systems and may be generic to be adapted to any simulation model. It corresponds to the $C_{communication}, X_{cext}, Y_{cext}$. This layer may be seen as a gateway when synchronisation and exchanges are done by another module (such as RTI *RunTime Infrastruture* in HLA simulation).
The second one is the **Interface layer** which fits data (interoperability). Its role is to format the data (received or to be sent) and ensure its consistency. Data exchanged between Model layer and outside are translated according to a generic format. So it avoids multiplying data types that induce a consistency verification complexity. Translation functions δ_{trans} and $C_{interop}$ realise these operations. The former may not affect inputs and outputs if external and

internal data are identical.
The third one, the **Model layer**, represents and describes model (to be simulated or a part of it) without taking into account the information about simulation implementation architecture. This layer is based on the component notion. It corresponds to $C, E_c, X_c, Y_c, \delta_{link}$. The encapsulated model is independent from its implementation architecture. Moreover it may be reused if considered as a component. This organisation is illustrated in the figure 5.

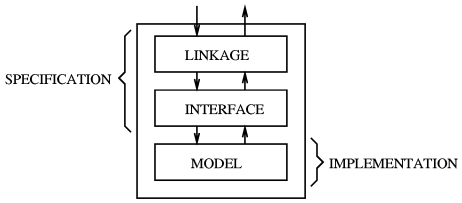


Figure 5: Internal Structure of a Petri Net Component

APPLICATION TO HLA

We modelled a problem of landing gear by using our approach of the PNC. The problem consists of simulating landing gears and checking if they open or close in a limited time. An order given by the pilot sends information to a centralised command. This command sends the appropriate order to gears and trapdoors.
A landing gear is made of a gear and a trapdoor. A plane has three landing gears (front, right and left) that is called the landing system. First the physical system of the gear and the trapdoor is modelled in two independent atomic PNCs. Then these two PNCs are coupled in a new coupled one. Coupling under shuffle has been used since the gear and the trapdoor have no common data. This coupled component is duplicated two times in order to have three landing-train components.
The centralised command has been modelled with a Petri Net which is included in the model layer of a new PNC. The Petri Net is given in the figure 6. This component sends information, more precisely orders to the three landing-gear components. This communication is made through specific places called communication places. There are two kinds of places: the source-places and the well-places. The source-places *receive* their tokens from other components whereas well-places *send* their tokens to other components. These two kinds of places are represented differently in the command model. Source-places are black and well-places are grey.
Once all the components are modelled, they are implemented on a standard architecture HLA that provides all the services to enable a simulation: communication ones, synchronisation ones and data handling ones. In this way, PNCs have to be fitted to these HLA services, as communication is made via RTI. The resulting global system is illustrated in figure 7.

The Linkage layer of each component provides the services needed for communication through RTI. In this aim, two interfaces are added in this layer. The first one called RTI-Interface translates services sent by RTI into functions understandable by the component. The second one called Component-Interface translates functions sent by the com-

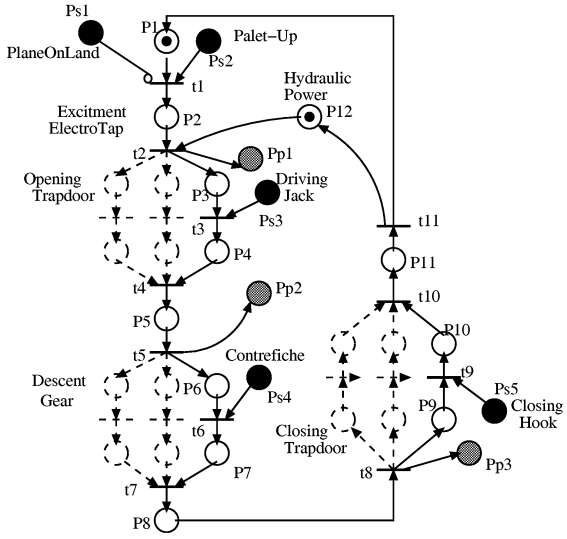


Figure 6: Command modelled with Petri Nets

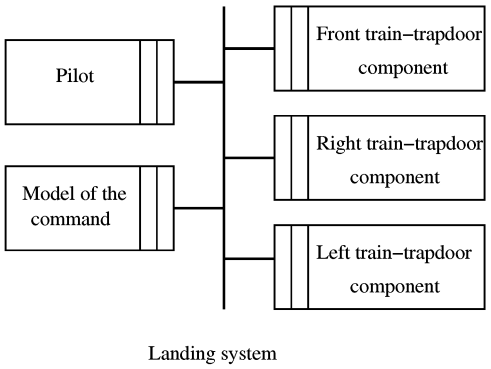


Figure 7: Landing system implemented on a HLA simulation

ponent into RTI understandable services. These two Interfaces promote interoperability between PNC and HLA. Interoperability may be briefly defined as the ability among heterogeneous systems to communicate. It means that system services have to match and exchanged data must have the same sense, same format, same language. This second point called the semantical interoperability may be achieved through the Interface layer.

Interoperability

Interoperability is an essential notion in distributed simulation and in modularity achievement. A very short definition is that it allows to exchange pertinent information and it enables communication between heterogeneous entities. As entities are different (because of the model for example) some conflicts may appear and cause communication problems.
We proposed in [Combettes and Nketsa, 2003] a way to analyse OMT that is the FOM and the SOMs. We are concerned here by building OMT tables.
We deal with an example here. We want to interconnect two atomic PNCs (PNC1 and PNC2) so we detail the necessary steps to build the resulting coupled PNC. Considered com-

ponents are the PNC of the Gear and of the Trapdoor from the landing system example. These PNCs are coupled in parallel as they have no input/output in common and they are supposed to be different. After having verified the structural condition PNC1 and PNC2 outputs (respec. inputs) are gathered in a global output (respec. input) set. By applying closure under shuffle condition , we deduce that:
$X_c = X_1^e \cup X_2^e$ and $Y_c = Y_1^e \cup Y_2^e$
(Note that "^{e}" stands for *external*). The semantical coupling condition doesn't exist because there is no relationship between the inputs (respectively outputs) of these two PNCs. Consequently the translation functions are equal to the Identity function. In the case where a semantical coupling does exist, the unsatisfaction of semantical coupling condition may be overcome through the translation functions. For example, as :
$\delta^1_{translInt-Ext} : Y_1 \longrightarrow Y_1^e$ and
$\delta^2_{translExt-Int} : X_2^e \longrightarrow X_2$,
$Y_1^e = X_2^e$ may be obtained by correctly adapting the translation function.
As linking may be done, δ_{link} function enables to inform whether a data is linked. Even in this case interoperability can not be completely achieved. In this aim, we propose a structure called $C_{structure}$ that is made of a set of tables contains all the data characteristics such as types, accuracy, precision etc ... Each PNC owns such a structure. Documented data are only the external ones. Moreover coupled data that become internal data of coupled PNC don't need to be specified in the coupled PNC structure. So interesting informations contained in the sub-PNCs are copied and included in the coupled PNC structure.
In order to be adaptable to HLA, we decided to use tables in our $C_{structure}$ having the same structure of OMT ones. However some tables and fields of OMT are not useful in our component approach. The useless table is the one concerning the routing space since routing space is not used in our component approach. Useless fields are the ones related to routing space and the ones concerning the instance attribute handling by the federate. The reason of this uselessness is that the data contained in these structures may be accessible to any component. To overcome this point we add an additional table that documents relationships between external data, the adapted external data and the translation function enabling this adaptation.
This reasoning may also be applied to other composition ways.
PNC coupling means interconnecting PNC descriptions and obtaining a specification of the whole system. In our approach, PNCs are linked through their ports. It is obvious that coupling conditions have to be verified and respected. Moreover in order to build resulting component, closure under coupling rules is used. Interconnecting PNCs implies data exchanges. As PNCs may be different, data exchange is allowed through $\delta_{translation}$ functions. This approach enables to detect language and semantical problems. $C_{interop}$ set is a table recording types and characteristics of all the data being exchanged with outside of PNC.

CONCLUSIONS

We have defined a component in this paper. We give formal definition of an atomic and a coupled component and demonstrate coupling rules. In our approach model body and the elements in relation with environment are separated as in [Michel and Wiels, 1997], however all the elements are gathered in only one layer (Interface). Moreover a communication layer containing communication primitives is included. As our approach makes the different layers independent, interoperability may be treated in an easier way. The future work is to exercise this approach in the context of an other tool [http://www.calytrix .com,] proposed by Calytrix which is an HLA development environment based component.

1. REFERENCES

Cenzis, Z. and Oguztuzun, H. (2002). A COM component set for HLA interface specification. In *SISO Fall 2002 Simulation Interoperability Workshop SIW'02*. Paper 02F-SIW-037.

Combettes, S. (2003). *Contribution à la définition d'un cadre de conception des systèmes virtuels par l'utilisation conjointe des Réseaux de Petri et de la simulation distribuée*. PhD thesis, Université P. Sabatier, Toulouse, France.

Combettes, S. and Nketsa, A. (2002). A virtual system approach for distributed simulation of complex systems. In *IEEE international Conference on Systems, Man and Cybernetics (IEEE-SMC'2002)*, Hammamet, Tunisia.

Combettes, S. and Nketsa, A. (2003). Using coloured petri nets to check interoperability in a HLA federation. In *European Simulation Interoperability Workshop (EUROSIW'02)*, Stockholm, Sweden. Paper 03E-SIW-084.

http://www.calytrix .com. http://www.calytrix .com. Web site.

Kiehn, A. (1989). Petri nets systems and their closure properties. In *Lectures Note in Computer Science - Advances in Petri Nets 424*, pages 306–328. Springer-Verlag.

Michel, P. and Wiels, V. (1997). A framework for modular specification and verification. In *Formal Methods Europe 1997*, Graz.

Radeski, A., Parr, S., Keith-Magee, R., and Wharington, J. (2002). Component-based development extensions to HLA. In *Proceedings of the Spring Simulation Interoperability Workshop*, Orlando, Floride. Paper 02S-SIW-046.

Zeigler, B., Ball, G., Lee, H. C. J., and Sarjoughian, H. (1999). Implementation of the DEVS formalism over the HLA/RTI : problems and solutions. In *Spring Simulation Interoperability Workshop*.

ENCAPSULATION IN AN OBJECT-ORIENTED NOTATION BASED ON MODULAR PETRI NETS

Cécile Bui Thanh
LACL – Université Paris 12
61, avenue du général de Gaulle
F-94010 Créteil, France
bui@univ-paris12.fr

Hanna Klaudel
LaMI – Université Evry-Val d'Essonne
523, Place des Terrasses
F-91000 Evry, France
klaudel@lami.univ-evry.fr

KEYWORDS
Object-orientation, encapsulation, coloured Petri nets, semantics.

ABSTRACT

In this paper, we address the problem of expressing encapsulation, *i.e.* access restriction levels (like private, public and protected) to attributes and methods, in a concurrent and object-oriented notation called BOON and provide it with a modular semantics in terms of high-level Petri nets. This feature allows to hide the fields of a class in order to prevent undesired change of the state of the class instances. The Petri net semantics gives a way to use automated verification and simulation tools on the translation of a system modeled using this object-oriented notation.

INTRODUCTION

In this paper, we propose a way to express encapsulation in an object-oriented modeling, in terms of Petri nets.

The starting point of our approach is a Basic Object-Oriented Notation (BOON) (Bui Thanh and Klaudel 2003), an extension of a high-level programming language, B(PN)^2 (Best and Hopkins 1993), which comprises most traditional concepts of parallel programming, like parallel composition, iteration, guarded commands, procedures and communications. BOON allows for defining classes and objects, and comprises single class inheritance, polymorphism and dynamic binding. Thanks to its simplicity, it can easily be used as a basis for various extensions and the results found for it may then be applied to "real-life" languages.

Another advantage of BOON is that it has already a concurrent formal semantics in terms of a class of high-level (coloured) Petri nets, called *M-nets* (Best et al. 1998, Fleischhack and Grahlmann 1997, Lilius and Pelz 1996, Klaudel 2001). The particularity of M-nets is that they are provided with a set of composition operations and allow to represent large (possibly infinite) systems in a compact and structured way. Moreover, they are implemented in the PEP toolkit (Grahlmann and Best 1996), allowing to simulate a modeled system and also to verify its properties *via* model checking.

In this paper we present an extension of BOON allowing for the encapsulation of the fields of a class, *i.e.*, the possibility to declare the fields with an access restriction level (private, protected, or public). Such an extension needs a deep reconsideration of the M-net semantics of object-oriented features, but it is necessary for preventing, *e.g.*, undesired access to an attribute of an object. Another advantage is that this gives the possibility to a translation of concurrent system specifications written in object-oriented formalisms or languages into Petri nets, and thus, to allow automated simulation or verification of the obtained Petri net using existing Petri-net based tools (Grahlmann and Best 1996, Aarhus 1993, Mäkelä 1999). This solution may be seen as an alternative to other Petri net based formalisms capable to express object-oriented concepts, which often use more complex net classes. This is the case, for instance, for *Object Petri Nets* (OPN) (Lakos 2001), whose nets are enriched with net tokens, or for CO-OPN (Biberstein et al.2001) and CLOWN (Chizzoni 1996), which use algebraic Petri nets (nets extended with algebraic data types).

SYNTAX AND INTUITIVE SEMANTICS OF BOON

Basic Object-Oriented Notation BOON (Bui Thanh and Klaudel 2003) is a parallel programming language comprising shared memory parallelism and object-oriented concepts of *classes*, *objects*, *inheritance*, *polymorphism* and *dynamic binding*. The Table 1 shows a fragment of the syntax of BOON (with keywords typeset in **bold** face, non-terminals in roman face and *italic* denoting values supplied by the program):

Table 1: A Fragment of the Syntax of BOON

program	::=	**program** cblock
cblock	::=	**begin** cdecl block **end** | block
block	::=	**begin** scope **end**
scope	::=	com | decl ; scope
com	::=	⟨expr⟩ | *c.m*(arglist)
		| block | (com) | **do** alt-set **od**
		| com ‖ com | com ; com

An atomic command is a BOON expression "⟨expr⟩", *i.e.*, a term constructed over operators, constants (from a given set $\mathbb{V}$), program identifiers (referring global variables or attributes), which can be executed if the expression evaluates to true. A variable v can appear in "expr" as $'v$ (pre-value) or v' (post-value), denoting respectively its value just before and just after performing the command, or just as v if the command does not change its value

A class is a high-level abstraction defined by a *name* in the set $\mathbb{C} = \{C, D, \ldots\}$, and a set of characteristic fields (*attributes*) and services fields (*methods*). An instance of a class C is an *object* of type C, which has its own identity and state (given by the values of its attributes) and contains all fields defining C. It can be created (resp. destructed) by calling a particular method defined in C, called a *constructor* (resp. a *destructor*). The declaration of a class is of the form **class** C { attdecl methdecl }, where "attdecl" and "methdecl" are the attribute and method declarations.

A class D may *inherit* from another class C, which means that D has all the fields of C, but it may *override* (and thus hide) some of them and may also have additional fields. The corresponding declaration contains then only the declarations of overridden and additional fields.

We allow *inclusion polymorphism*, which means that an object variable c declared of type C may also refer (at a stage of the execution) to an object of any subclass of C. An attribute a of c may appear in an expression as $c.a$, referring to the attribute a of the class of c, or the closest superclass of it in the inheritance tree, determined dynamically at the execution. The latter mechanism is called *dynamic binding* and exists also for a method calling.

The keyword **this** can be used in the body of a method of a class C for referring to the object to which the method is applied. So, in the body of a method m of class C, an attribute a may appear as **this**.a (and analogously for a method m' of C), which has the same meaning as if **this** was an object variable. The keyword **this** may also be used in an operation at the expression level. Also, the keyword **super** can be used in the body of a method of a class C for referring to a field of the superclass of C. It can also be juxtaposed k times, which would refer to the k^{th} parent of C. For instance, **super.super.**a refers to the attribute a of the parent of the parent class of the object to which the method is applied.

In this paper, we introduce in BOON encapsulation features in order to control the access to the class fields, by adding privateness keywords before each field declaration. In a class C, the keyword **public** makes a field accessible from anywhere, whereas **private** restricts the access to methods of C and forbids to override it, and **protected** is like **private** but allows also the access to methods of subclasses of C and to override it. The privateness of a field is preserved through inheritance, and if a subclass overrides a field, the privateness of this field cannot be changed.

M-NETS AND M-NET SEMANTICS OF OBJECT-ORIENTED CONCEPTS IN BOON

M-Nets

M-nets (Best et al. 1998) form a class of high-level (coloured) Petri nets provided with a set of operations giving to them an algebraic structure. Like other high-level Petri net models, M-nets carry the usual annotations on places (sets of allowed tokens), arcs (multisets of annotations) and transitions (guards, that is, a Boolean expression which plays the role of an occurrence condition). In addition, places have a status (entry, exit or internal) used for net compositions; transitions carry labels used for inter-process communications, which are similar to CCS ones (Milner 1989) but extended to (multi)sets of actions with arbitrary arity. The communications can be enforced using the operations of *synchronisation* and *restriction* w.r.t. a set of actions. For instance, the synchronisation w.r.t. $\{act\}$ applied to a net containing the transitions with labels $\{act(x, y), term(y)\}$ and $\{\widehat{act}(u, 5)\}$ will produce in the net a new transition labelled $\{term(5)\}$, obtained by gluing the two former transitions together, while the restriction will remove from the resulting net the transitions whose labels involve act or $\widehat{act}$.

The marking of an M-net associates to each place a multiset of values (tokens) from the type of the place and the transition rule is like for other high-level nets. The initial marking corresponds to one black token in each entry place of an M-net and no token elsewhere.

M-net Semantics Of BOON

Intuitively, the M-net semantics of an object-oriented program, defined in (Bui Thanh and Klaudel 2003), involves three parts: the class declarations, class instances management and the main command. All these parts (M-nets) are put in parallel, synchronised and restricted w.r.t. all communication actions, as sketched in the figure 1.

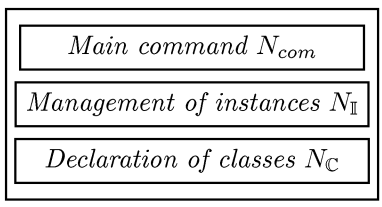


Figure 1: Structure of the M-net Translation of a BOON Program.

Each object (instance of a class) is uniquely identified by an identifier $id \in \mathbb{I}$, which can be considered as a pointer, and has also a *type* κ, which may differ at a stage of the execution from the declaration, like in the scope "**ovar** $c : C \quad \langle c' = \mathbf{new}_D(\text{initlist}) \rangle$". The value of an object c is thus the pair (id_c, κ_c). Such values are managed by the *class instances* M-net $N_{\mathbb{I}}$, which provides a free identifier from the set $\mathbb{I}$ to each new object, keeps its actual type, and gets back each identifier released by a destruction.

The part $N_{\mathbb{C}}$ for the class declarations is composed of a parallel composition of one *inheritance directory* M-net and an M-net for each class declaration.

The *inheritance directory* M-net $N_{H(\mathbb{F})}$ stores during the execution of the program the set of tokens $\mathbb{F} = \{(D, C) \mid C \text{ is the superclass of } D\}$ which encodes the class inheritance tree. It allows for accessing to an attribute or a method of the superclass (keyword **super**).

The M-net semantics of a declaration of a class C is mainly composed of an M-net for each field of C. Each of them has again two or three parts: a *resource* M-net, an *internal interface* M-net and, if the field is public, an *external interface* M-net. In order to handle properly the initialisation and the destruction of the attributes at the creation or destruction of an object, the semantics of a class declaration includes also an *instantiation* and a *destruction* M-net. Moreover, if a class D inherits from a class C, then the semantics of D contains an additional part composed of *internal request* M-nets and *external request* M-nets. See also the figure 2.

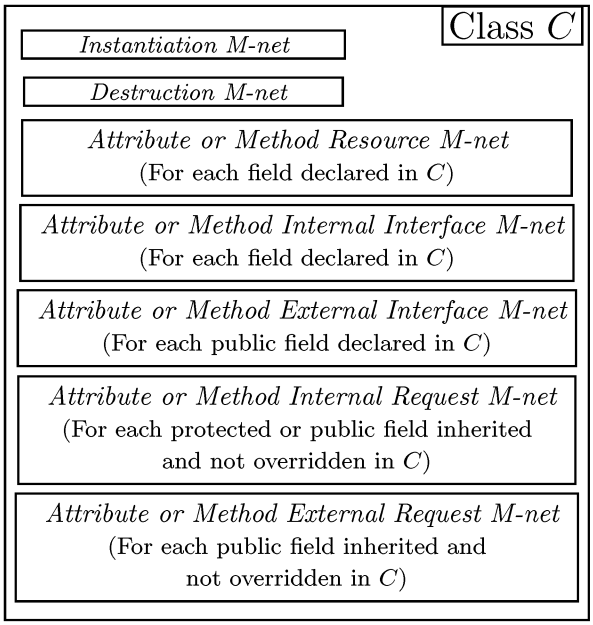


Figure 2: Structure of the M-net of an Object

The resource M-net of an attribute a keeps the current value of a for each instance of the class where a is declared. If a is a standard attribute, this value belongs to a set $V \subseteq \mathbb{V}$ being the type of a, and if a is an object attribute, it is a pair (id, κ), where id is the identifier of the object of type κ assigned to a. For instance, if a is an object attribute belonging to a class C, then for each object c of type C identified by id_c, the resource M-net of a keeps the token $(id_c, (id, \kappa))$, where (id, κ) is the value of the object assigned to $c.a$. When an object is created, the *instantiation* M-net asks for a fresh identifier id_c for it and initialises all the attributes associated to this identifier by putting the corresponding tokens in the resource M-nets of the attributes. An attribute a of the object identified by id_c can then be updated through an action $access_a(id_c, x, y)$, where x is the value of the attribute before the access, and y is the value after the access. An object attribute oa may also be explicitly destructed (independently of its owner object) through an action del_oa. Finally, when an object of type C is destructed, its associated attributes are destructed too. The *destruction* M-net of the class C triggers the removing of all the tokens corresponding to the destructed object from the attribute resource M-nets. The resource M-net of a method is more complex but is used in a similar way.

The *external interface* and *request* M-nets for a field a (attribute) or m (method) of a class C handles any "external" access request to this field through actions of the form $ereq_a(\ldots)$ or $ereq_m(\ldots)$ (this does not concern requests sent from a method of C or a subclass of C) and relays it to the corresponding internal interface or request M-net. Their presence means that the corresponding field is public. The *internal interface and request* M-nets for a field a or m of a class C handle the dynamic binding and the access to the resource M-nets. They receive the access requests through actions of the form $req_a(\ldots)$ or $req_m(\ldots)$. If a field of C is private, there is no request M-net, nor external interface M-net for it, which disables the access to it if the request does not come from a method declared in C.

The part N_{com} for the main command is represented by the M-net of the corresponding block. The main idea in describing a block is (i) to juxtapose the nets for its local variable declarations with the net for its command followed by a termination net for the declared variables, (ii) to synchronize all matching data/command transitions and to restrict these transitions in order to make local variables invisible outside the block and (iii) to add the initial marking to the obtained net (typically a black token $\bullet$ in each initial place).

Each variable or method declaration is translated into a corresponding *resource M-net*. For instance, the declaration of a (global) standard variable v of the type $V \subseteq \mathbb{V}$ gives rise to the M-net $N_{v,V}$ represented on the top in the figure 3. The current value of the variable v is stored in the place of type V and may be updated using the $\{\widehat{access_v}(x, y)\}$-labelled transition. A command enforces the change of the value of v through the syn-

chronisation mechanism w.r.t. the action $access_v(...)$. The resource M-net for an object variable is similar but has also a transition to handle the explicit destruction of the object assigned to the variable.

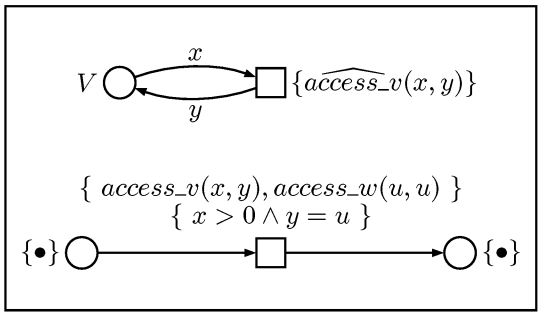


Figure 3: The Resource M-net of a Standard Variable and an Example of M-net Reading its Value.

Sequential and parallel compositions are directly translated into the corresponding net operations, *e.g.*, $\mathsf{Mnet}(com_1; com_2) = \mathsf{Mnet}(com_1); \mathsf{Mnet}(com_2)$, where Mnet is the semantical function used for the translation, while the semantics of the "**do** ... **od**" construct involves the M-net iteration operator (not explained here). The semantics of an atomic command "⟨expr⟩" is the M-net $N_{\alpha\gamma}$ where α is a set of actions corresponding to the access to the program variables involved in "expr", and γ is the guard obtained from "expr" with program variables appropriately replaced by net variables, like *e.g.*, x for $'v$, y for v' and u for w in: $\mathsf{Mnet}(\langle 'v > 0 \wedge v' = w \rangle) = N_{\{access_v(x,y), access_w(u,u)\}\{x>0 \wedge y=u\}}$. This M-net has one transition as shown on the bottom in the figure 3.

Method calls and attribute accesses for an object are modeled in almost the same way, since the corresponding interface M-nets are very similar. In both cases, we need to know the value (id, κ) of the object and use it in a request for the corresponding method or attribute. If the access is performed by a method of the class owning the requested field, *i.e.* through a command of the form "**this.**...", then the request is sent with an action of the form $req_a(...)$ or $req_m(...)$. If the access is done through an expression of the form "$(\textbf{super.})^k$ ...", then the request action is the same, but is associated to a request to the inheritance directory M-net (in order to know the class to which the request must be sent) and an action testing if the field is not private. Otherwise, the request action is of the form $ereq_m(...)$ or $ereq_a(...)$. For instance, the transition label (lab) of the translation $N_{(lab)}$ of the atomic BOON command $\langle\ 'c.sa > 5\ \rangle$, which tests wether the attribute sa of c is greater than 5, becomes $\{access_c((id_c, \kappa_c), (id_c, \kappa_c)), \widehat{ereq_sa}(id_c, \kappa_c, x, x)\}\{x > 5\}$.

The call of method m is translated to a *call* M-net similar to a procedure invocation (Klaudel 2001) (not explained here).

CONCLUSION

We extended the object-oriented notation BOON, with encapsulation features and defined for it a fully compositional Petri net semantics in terms of M-nets. This led to the development of a representation for classes, objects, attributes and methods supporting single inheritance, dynamic bindings and access restriction levels. This approach may easily be extended in order to allow for more polymorphism, for instance, by accepting several user-defined constructors or different methods having the same name. The former can be modeled in our approach using customised instantiation M-nets, while the latter can be obtained by simply preprocessing the homonymous methods in order to give them distinct names.

Our future works will address the application of M-nets to the translation of UML (Booch et al.) diagrams and the modelling of agent-oriented systems.

REFERENCES

E. Best and R. P. Hopkins. 1993. "B(PN)^2 – a Basic Petri Net Programming Notation." *PARLE'93*, LNCS 694. Springer.

E. Best, W. Fraczak, R.P. Hopkins, H. Klaudel and E. Pelz. 1998. "M-nets: an Algebra of High-Level Petri Nets, with an application to the semantics of concurrent programming languages." *Acta Informatica 35*. Springer.

O. Biberstein, D. Buchs and N. Guelfi. 2001. "Object-Oriented Nets with Algebraic Specifications: The CO-OPN/2 formalism." APN, LNCS 2001. Springer.

G. Booch, I. Jacobson, and J. Rumbaugh. 1998. "The Unified Modeling Language User Guide." Addison Wesley.

C. Bui Thanh and H. Klaudel. 2003. "Object-oriented modelling with high-level modular Petri nets." Technical report 2003-03, University of Paris 12, LACL.

A. Chizzoni. 1996. "CLOWN: CLass Orientation With Nets." Master degree thesis, Univ. of Milan.

H. Fleischhack and B. Grahlmann. 1997. "A Petri Net Semantics for B(PN)^2 with Procedures." *PDSE 1997*, IEEE Computer Society, Boston.

B. Grahlmann, and E. Best. 1996. "PEP – More than a Petri Net Tool." *TACAS'96*, LNCS 1055, Springer.

H. Klaudel. 2001. "Compositional High-Level Petri Net Semantics of a Parallel Programming Language with Procedures." *Science of Computer Programming 41*. Elsevier.

C. Lakos. 2001. "Object Oriented Modelling with Object Petri Nets." APN, LNCS 2001. Springer.

J. Lilius and E. Pelz. 1996. "An M-net Semantics for B(PN)^2 with Procedures." *Proc. of ISCIS'96*, Volume I, Middle East Technical University.

M. Mäkelä. 1999. "MARIA: modular reachability analyser for algebraic system nets." Online manual, http://www.tcs.hut.fi/maria.

R. Milner. 1989. "Communication and Concurrency." Prentice Hall.

University of Aarhus. 1993. "Design/CPN Reference Manual for X-Windows." Online manual, http://www.daimi.aau.dk/designCPN/.

AUTHOR BIOGRAPHY

CECILE BUI THANH obtained a DEA/MSc in Computing Science from the University of Paris XI at Orsay in 2001. She joined then the the Laboratory of Theoretical Computer Science (LACL) of the University Paris XII as a PhD student, and supervised by Dr. Hanna Klaudel. She is working on object-oriented modelling in concurrent systems.

DR. HANNA KLAUDEL obtained a DEA/MSc in Computing Science from the University of Paris XI at Orsay in 1992. In 1995 she obtained a PhD from Laboratory of Computing Science (LRI) of the University of Paris XI, for work on compositional modelling of concurrent systems and programming languages. In 1996 Dr. Klaudel joined then the Laboratory of Theoretical Computer Science (LACL) of the University Paris XII at Créteil, first as a Junior Lecturer, then a Lecturer. In 2000 she obtained a Habilitation from the University Paris XII at Créteil for the work on structured modelling of complex concurrent systems. Recently, in 2003, she joined the Computer Science Laboratory (LaMI) of the University of Evry as a Professor. During last few years Dr. Klaudel has been involved in several research projects, in particular, the Esprit WG Caliban and bilateral French German projects, and took part twice in programme committee of international conferences.

USING META-MODELLING TO PROCESS PETRI NETS MODELS OF SUPPLY CHAINS

Hendrik Van Landeghem and Carmen-Veronica Bobeanu

Ghent University, Department Industrial Management, Technologiepark 903,
B-9052 Ghent, Belgium
e-mail: hendrik.vanlandeghem@ UGent.be
Carmen.Bobeanu@UGent.be

KEYWORDS
Petri nets, structural modelling, supply chain, meta-modelling

ABSTRACT

This paper addresses the authors' current results to generate a custom tool supporting the construction of simulation models of complex supply chains. The concepts proposed in the authors' previous developments addressing the modular synthesis of Petri nets-based simulation models of complex discrete event systems are implemented in AToM^3, A Tool for Multi-formalism, Meta-Modelling. The ultimate goal of the research considered in this presentation is to provide a natural, simple and powerful method for describing and analysing the flow of information and material in supply chains not far from the industrial engineers' habitual notions of system design and operation.

1. INTRODUCTION

Nowadays, simulation becomes one of the top technologies used by the industrial engineers, operations researchers and management scientists. Although simulation is widely accepted as an indispensable problem solving methodology, Banks (Banks 2001) notes that "it is used in only a small fraction of the cases where it can have a significant impact". The solution suggested to accelerate the growth in simulation use is the reduction of the complexity of using the above tool. Moreover, as remarked in (Vangheluwe et al. 2001), to tackle problems of ever increasing complexity of today's highly competitive systems, "modelling and simulation research is shifting from simulation techniques to modelling methodology and technology".
It is the purpose of the authors' work to bring forth ideas and provide tools able to limit or even eliminate some drawbacks as they appear in the analysis of current practices in this field, addressing the construction of simulation models in the framework provided by a widely employed specialized formalism: Petri nets (PN). The main strength of the proposed approach lies in its ability to develop systematic PN-based representations, replacing the intuitive construction of PN models. Thus an important step is made towards the automated generation of re-usable PN-based simulation models.
Using meta-modelling, a custom tool supporting the models processing according to the proposed method is developed. Thus the proposed formalism can be further tailored to specific needs by modifying the meta-model.
This paper illustrates the application of the proposed approach to the structural modelling of supply chains (SCs), with a final aim to encourage the use of simulation-based methods as a key factor which supports the representation and analysis of (complex) SCs.

2. AN INCREMENTAL APPROACH TO MODELLING SUPPLY CHAINS USING PN

In this Section several ideas underlying our earlier developments (Van Landeghem and Bobeanu 2002) in SCs representation in the PN domain are further elaborated.
We recall that the central idea of our research is a refined PN representation in terms of sets and operations on sets, focusing on the main attributes of a composite PN object specification and anticipating the application of Zeigler's theory (Zeigler et al. 2000) in the PN domain. We recall as well the key elements supporting the manipulation of these abstractions introduced in our previous work (Bobeanu and Alla 1998, Kerckhoffs and Bobeanu 2001, Van Landeghem and Bobeanu 2002):

- *primitive/atomic model*: block component with a well-defined interface, that can be seen as a process with an input transition, T_i, supplying the input of its activity and an output transition, T_o, enabling its evacuation and a place P modelling the temporal entities advance in the net (production place) or constraints to be satisfied (synchronization place);
- *coupling templates*: standardised means to couple building block components;
- *step-by-step procedure for the construction of compound PN-models*: systematic bottom-up construction of PN models using as input information (about e.g. primitive system components, entity flows, routing constructs, etc.) gathered from the top-down system analysis.

One of the new ideas considered in (Van Landeghem and Bobeanu 2002) is that the development of a well-grounded set of coupling templates forces one to partition the entity flows involved in a SC into three distinct types:

- orders placed by the customer (denoted as *Ea*);
- internal orders (denoted as *Eb*);
- products (denoted as *Ec*).

The formal developments proposed at this stage as a result of this refinement broaden the applicability of the authors' contribution to also address the interfacing questions of different flows of entities. In this perspective the following rules are introduced:

I. The **internal coupling templates** of two **sequential models** *A* and *B* will allow a fusion to be applied to a pair of transitions (T_o^a, T_i^b), with T_o^a the output transition of the model A and T_i^b the input transition of the model B, provided one of the following cases applies:

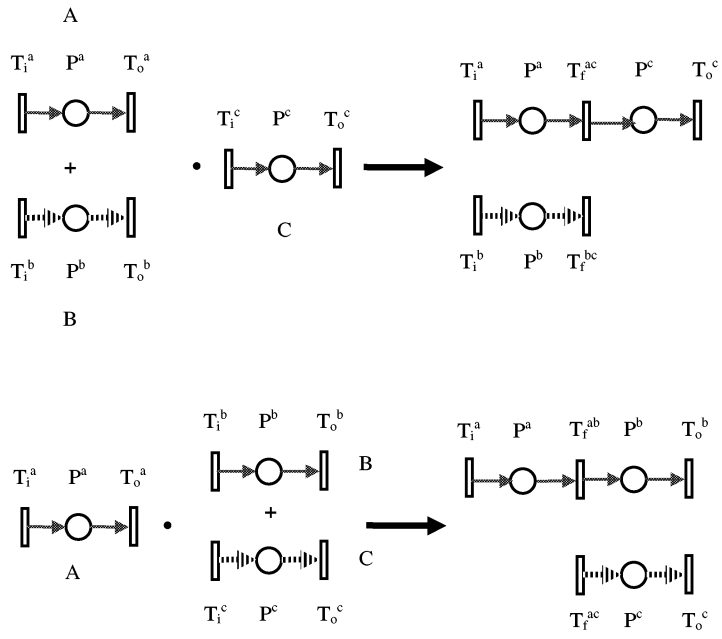


Figure 1. Templates allowing Combinations of Basic Coupling Rules for Different Flows of Entities

I.1 The above transitions are enabled for the same type of entities (see Figure 1);

I.2 A transformation is allowed between the types of entities enabling the candidate transitions for applying a fusion rule (such as a transition enabled by a component of the information flow and a transition generating a component of the material flow as a result of its firing). In Section 5 we will show an example of transitions fusion matching this (synchronization place) rule.

II. A **representative system of the input/ output transitions sets** of a coupled model to be used by an internal coupling while modelling synchronization and parallelism, will be structured according to the types of entities (or classes identified within the scope of a given type) enabling the addressed input/ output ports. The induced structure will prevent disjoint classes of entities to be supplied/ evacuated using the same input/output ports of the model (see Section 5).

3. USING META-MODELLING TO MODEL FORMALISMS

Meta-modelling is the process of designing modelling languages through meta-notations. A *meta-model* is a precise definition of the constructs and rules needed for creating semantic models in a given modelling environment. If we consider a modelling language as a concrete representation of a formalism, a *meta-model* is a model of the formalism itself.

Meta-models specify the *syntax* (the elements of the language) and *semantics* (the meaning) of a class of models.

Such a meta-specification allows the automatic generation of a domain-specific modelling and simulation environment to process (create, edit, check, optimize, transform) models described in the specified formalism. The concepts proposed in our developments addressing the modular synthesis of PN-based simulation models of complex discrete event systems are implemented in AToM^3, A Tool for Multi-formalism, Meta-Modelling (see AToM^3 home page). The main features of AToM^3 could be summarized as follows:

– "Languages" are graphs and each graph is implemented as a class that inherits from another graph: Abstract Syntax Graph (ASG).

– A "language" described by a graph is called a formalism.

– A graph hierarchy is considered where each formalism of a lower level is described in term of the next higher level.

– The meta-modelling levels considered by the tool are (de Lara and Vangheluwe 2002):

  ▪ *meta-meta-model*: description of the highest formalism - Entity-Relationship (ER) - in itself;

  ▪ *meta-model*: description of a new formalism (class of models) under the rules of the ER meta-meta-model;

  ▪ *model*: description of an object under the rules of a certain meta-model.

– Provides a GUI where graphs are created in ER formalism.

– Uses the scripting language Python and Object Constraint Language (OCL) as constraint languages.

– Parses meta-models in ER and generates code for a modelling environment where a user can create valid models.

In AToM^3, the important attributes of a meta-model inherited from the ASG class are:

– the name of the meta-model;
– a pointer to the first object created in the model;
– a list of strings with the allowed node types;

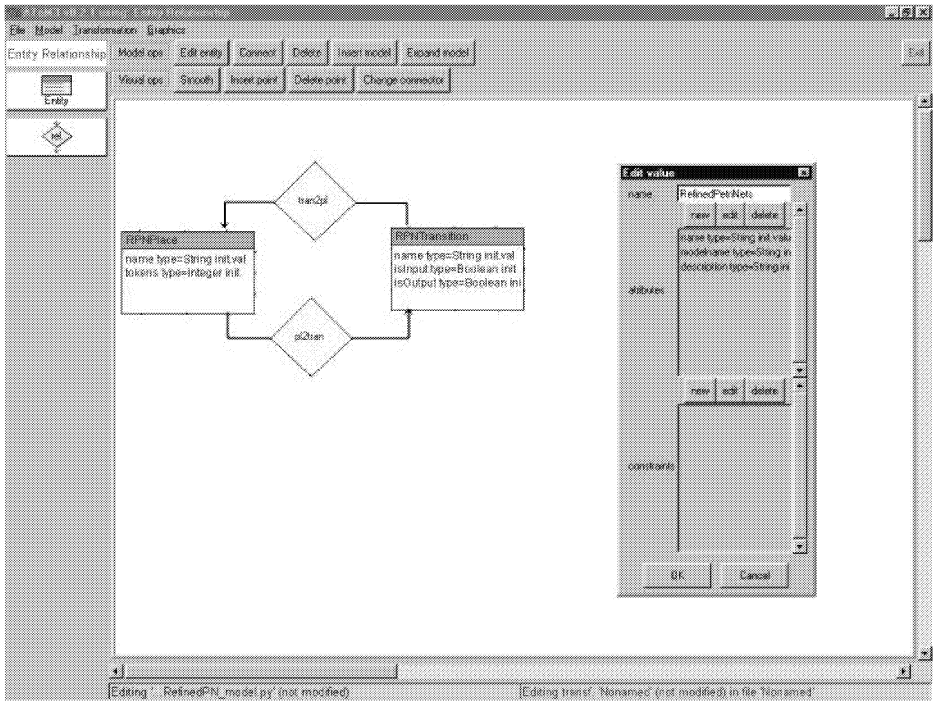


Figure 2. Meta-model of the Refined Petri Nets Formalism. Model Properties

– a dictionary indexed by the allowed node types, which contains lists of the current nodes in the model.

Each entity/ relationship that may appear in a model can be specified together with the following information:

– *Cardinality constraints*: define how an entity can be connected to a relationship and vice-versa.

– *Attribute*: the attributes an entity/ relationship will have in the generated formalism.

– Python-coded *constraints*: user defined constraints on the generated formalism.

– *Appearance*: graphical appearance of each entity/relationship when instantiated at the lower meta-level allowing as well to specify how some semantics attributes are displayed graphically.

While the models are represented internally in AToM^3 using ASG, model processing is described at a meta-level by means of graph grammar models.
Graph Grammars transform a graph representation into another. A Graph Grammar consists of several rules/productions. A rule matches a graph pattern in a given host graph and replaces it by another. A transformation is completed when the graph rewriting system cannot execute anymore rule.

4. CUSTOM TOOL FOR REFINED PN FORMALISM

We will present in the sequel the main steps in the generation of the Refined Petri Nets formalism from a model designed in the ER formalism, using AToM^3. As a first step, the model attributes were settled up (Figure 2). We emphasize on the importance of the name property since this is the name of the generated file for the user-defined formalism.
At the next step, the entities of our meta-model were defined: places (identified as RPNPlace in the meta-model description) and transition (identified as RPNTransition in the meta-model description) by specifying their properties.
The refinement of the transitions set envisaged in our formalism was implemented (Figure 3) by adding two Booleans (isInput and isOutput) to the transition entity, allowing the user to specify if the current transition is an input or output port, respectively, of the model. The

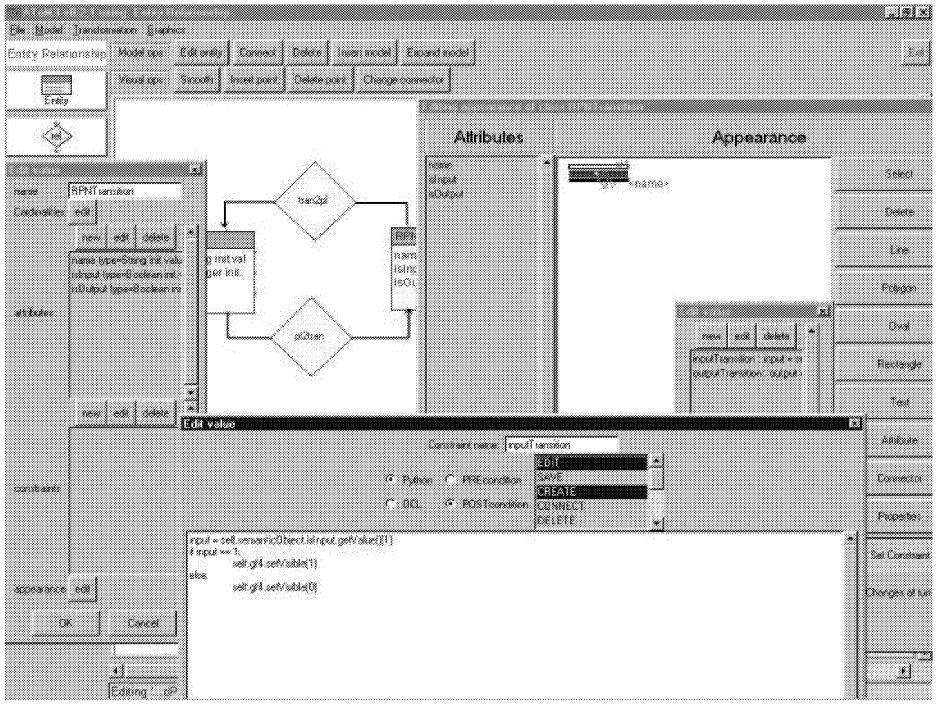


Figure 3. Setting Up the Properties of a Transition Entity

graphical expression of the transition entity was specified by the property Appearance, where different graphical objects were envisaged for a transition depending if it is input or output transition. In order to modify the appearance of the transition entity depending if the above-mentioned Boolean variables are set to TRUE or to FALSE, we have also added two Python coded graphical constraints to be evaluated after (*POSTconditions*) the events EDIT and CREATE.

In order to achieve the description of our meta-model, the two types of relationships established between the net components were defined: the arcs from a place to a transition (denoted as pl2tran components of the meta-model) and from a transition to a place (denoted as tran2pl components of the meta-model).

When generating code for the model with the properties described in Figure 2, several Python files were created:

- *RefinedPetriNets.py*: can be loaded in AToM^3 as a meta-model or as a model (in the *Buttons* formalism) of the user interface to be loaded with the formalism;
- *RefinedPetriNets_MM.py*: when the meta-model *RefinedPetriNets* is loaded, the functions defined in this file are added to the AToM^3 class methods.
- *ASG_RefinedPetriNets.py*: the graph that stores the description of the new formalism.

The tool generated in AToM^3 according to the above meta-specifications can be used to build, load and save refined representations of PN models proposed in our approach. Current developments are aimed to enrich this functionality by addressing the following types of model manipulations:

– Simulator specification: express the operational semantics of the refined formalism used by our method;
– Compound models synthesis: allow the implementation of the rules underlying the handling of the routing constructs identified by our method.

In this respect, our current work is focused on the description at a meta-level of the above-mentioned classes of model transformations by means of models in the Graph Grammar formalism.

5. EXAMPLE

Using the well-known example of the Beer Game (Simchi-Levi et al. 2000), we illustrate in the sequel how the proposed stepwise building process of PN models helps the user to better appreciate the nonlinearities in the SC and the logical relationship between different control structures revealed by the analysis of the informational and material flow. Figure 4 and Figure 5 show the generated tool supporting the synthesis of Refined PN models at work.

As noticed in (Van Landeghem and Bobeanu 2002), a similar structure as well as similar principles underlie the behaviour of each echelon of the Beer Game. This will of course result in similar constructions of the models associated to each component of the SC. The exception to this uniformity is the Factory, where the following principle does not apply: "A supplier can only fill an order if it has the necessary inventory". As there is no production capacity limit at this level, the factory's order is always filled in its entirety after the appropriate delay. The presentation below will only refer to the PN-based model of the Factory.

The structural modelling of the factory implies the following step-by-step procedure:

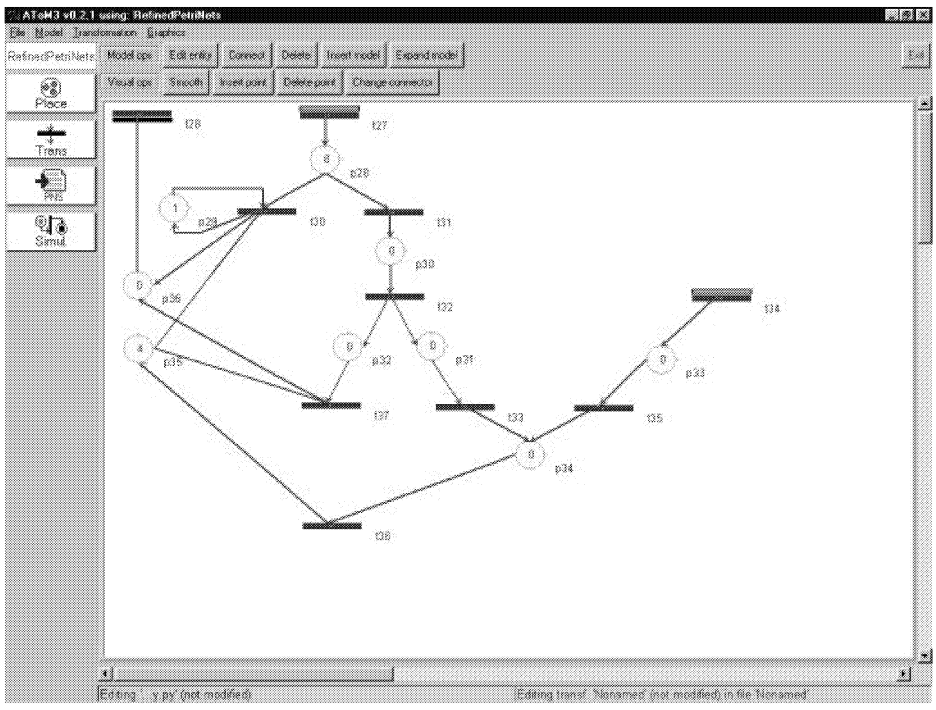


Figure 4. Refined PN-based Model of the Factory

Step 1. 1a) Nine **basic system components** (BSCs) are revealed by the analysis of the factory component:

- Entity *Ex* arrival, with $x \in \{a,b\}$;
- Orders fulfilment;
- Remaining orders processing;
- Backlogs record;
- Backlogs accumulation;
- Local production decision process;
- Storing entity *Ec* in the local inventory;
- Orders shipping.

Their corresponding **basic model components** (BMCs) are further represented using the refined PN models proposed by the authors. In order to illustrate the result of applying this step, we refer to, for instance, the BSC representing the "Backlogs accumulation". The PN representation of this BMC is represented by the place p32, the

Table 1. RCs Involved in the Entities Flow at the Factory

ROUTING CONSTRUCT	THE REAL-SYSTEM COMPONENT ADDRESSED BY THE ENVISAGED ROUTING CONSTRUCT	PRIORITY
Sequence	- Orders fulfilment activation process.	1
	- Local production decision process followed by storing entity *Ec* in the local inventory.	1
Conflict	Entity *Ea* arrival followed by orders fulfilment and remaining orders processing.	5
Accumulation	Backlogs record and entity *Eb* arrival, followed by local production decision process.	2
Parallelism	Remaining orders processing followed by backlogs accumulation and backlogs record executed in parallel.	3
Synchronization	The conflict induced by orders fulfilment and backlogs processing, and storing entity *Ec* in the local inventory, followed by orders shipping.	4

transitions *t32* and *t37* (with *t32* the input transition of the BMC and *t37* the output transition), and the arcs (*t32*, *p32*) and (*p32*, *t37*) in Figure 4. We can model all the above-identified BSCs in a similar way.

1b) Then the construction moves on to the abstract level and relates an **experimental frame** (EF) to every BMC constructed at this Step. In order to illustrate the results of applying this procedure, we extend, for instance, the above-constructed BMC representing "Backlogs accumulation". We relate an empty structure as the generator, G^{32}, defining arrival processes of entities of type *a*, to the input transition *t32*. A transducer, T^{32}, supporting statistics gathering as well as a tautology as an acceptor A^{32}, to the production place *p32*.

2a) Six system's **routing constructs** (RCs) are then identified following the flow of entities in the factory. They are listed in Table 1 together with their associated priorities according to logical inter-relationships. Below we will explain how to handle the above-identified RCs and their associated PN model components resulting in new model components.

As recommended by our method, we start with the construction of the compound models (CMs) associated to the two sequences revealed by the analysis of the factory (priority 1), followed by the construction of the CM associated to the accumulation (priority 2). Now we are in the position to initiate the construction of the component "Remaining orders processing followed by backlogs accumulation and backlogs record executed in parallel".

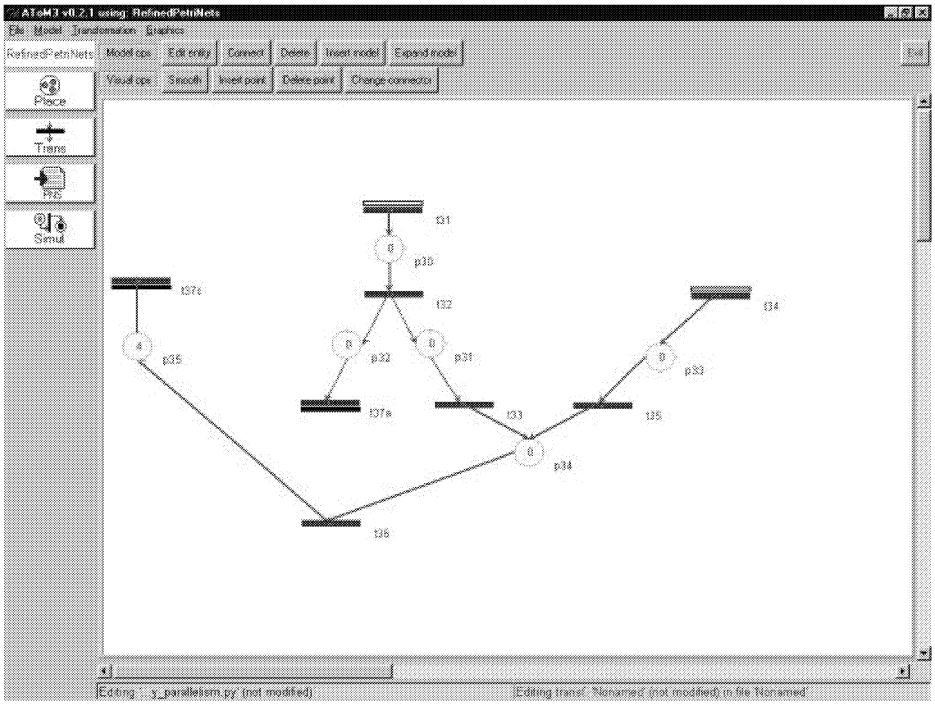


Figure 5. Handling Parallelism in the Synthesis of the PN Model of the Factory

Following the flow between the underlying BSCs revealed by the analysis of "Remaining orders processing followed by backlogs accumulation and backlogs record executed in parallel", it is easy to see that the case "parallelism" applies to the BMCs addressed by this RC. The PN-based representation of the corresponding CM is represented by the places *p30*, *p31*, *p32*, *p33*, *p34* and *p35*, the transitions *t31* (with *t31* the input transition of the CM, supplying entities of type *a*), *t32*, *t33*, *t34* (with *t34* the input transition of the CM, supplying entities of type *b*), *t35*, *t36*, *t37a*, *t37c* (with *t37a*, *t37c* the output transitions of the CM, evacuating entities of type *a* and *c*, respectively), and the arcs (*t31*,

p30), (*p30*, *t32*), (*t32*, *p31*), (*p31*, *t33*), (*t33*, *p34*), (*t34*, *p33*), (*p33*, *t35*), (*t35*, *p34*), (*p34*, *t36*), (*t36*, *p35*), (*p35*, *t37c*), (*t32*, *p32*) and (*p32*, *t37a*) in Figure 5.
The construction of the overall model follows in a similar way, according to the proposed method, and the priorities indicated in Table 1.
A special note should be made on the implementation of the formal developments presented in Section 2:

I. The refined internal coupling rule of two sequential models is used in handling the RC "Remaining orders processing followed by backlogs accumulation and backlogs record and entity *Eb* arrival, followed by local production decision process executed in parallel" (which normally should have asked for a fusion to be applied to the pair of transitions (*t32*, *t34*) (see Figure 4)).
II. A structured representative system of the output transitions set of a coupled model to be used by an internal coupling, is illustrated by the transitions *t30* and *t37* in Figure 4, used in the representation of the RC "Orders fulfilment and the sequence backlogs processing and storing entity *Ec* in the local inventory, followed by orders shipping". This is induced by the classes identified within the scope of the entity type *a*: fulfilled orders and backorders.

2b) Following each iteration undertaken at the structural level, the construction moves on to the abstract level. For each model component generated in this Step, an overall EF is assembled from the EFs related to the model components underlying the RC concerned, according to the rules proposed by the authors.

For example, the coupled model associated with "Remaining orders processing followed by backlogs accumulation and backlogs record executed in parallel" accommodates an EF with a generator, $\{G^{31}, G^{34}\}$, defining arrival processes of entities of type *a*, to the input transition *t31* and of entities of type *b*, to the input transition *t34*. Note that an empty structure is related as the generator G^{31}. The structure $T = T^{30} \cup \{T^{31} \cup T^{34} \cup T^{35}, T^{33} \cup T^{34} \cup T^{35}\} \cup T^{32}$ is the result of the modular synthesis of the transducer component, and a tautology makes the acceptor component of the above EF.
The overall system description in terms of PN achieved by applying the proposed method is shown in Figure 4.
Through self-containedness with input/output ports, the design of the overall model supports re-usability. The overall CM accommodates the EF specified by the following components:

- a Generator:

  $G = \{G^{27}, G^{34}\}$,

- an Acceptor

  A= .TRUE.

- a Transducer

  $T = T^{28} \cup T^{29} \cup T^{30} \cup \{T^{31} \cup T^{34} \cup T^{35}, T^{33} \cup T^{34} \cup T^{35}\} \cup$

  $\cup T^{32} \cup T^{36}$

The model of the Retail Store was described in (Van Landeghem and Bobeanu 2002). Similarly, the models of the warehouse and the distribution center can be generated. Finally, the overall model (featuring 36 places components and 37 transitions components) of the Beer Game is easily obtained by assembling the four above-mentioned PN model components in a *sequence* of length 4.

6. CONCLUSIONS

In this paper a systematic approach of the structural modelling of SCs is proposed, with a final aim to encourage the use of simulation-based methods as a key factor which supports the representation and analysis of (complex) SCs. Using meta-modelling, a custom tool supporting the implementation of the concepts and the generic method proposed by the authors, is currently under development with a final aim to support the automated generation of the PN models of SCs in a bottom-up approach.

REFERENCES

AToM^3 home page: http://atom3.cs.mcgill.ca/

Banks, J. "The Future of Simulation". 2001. In: Dolgui, A., Vernadat, F. (Eds.) Conception, analyse et gestion des systèmes industriels, Volume 1, pp. 11-12. *Actes de la troisième Conférence Francophone de Modélisation et Simulation (MOSIM'01)*, Troyes, France, 25-27 avr.

Bobeanu, C.V. and Alla, H. 1998. "Modelling Basic Discrete Event Systems - A Key Factor in Supporting the Representation and Analysis of Complex Production Systems". In: Z. Binder, B.E. Hirsch and L.M. Aguilera (Eds.), Management and Control of Production and Logistics. Proceedings of the IFAC /IFIP Conference MCPL'97, vol. 1, pp. 193-198, Campinas -SP-Brazil, Aug. 31 - Sep. 3, 1997. Oxford: Pergamon Publ.

de Lara, J. and Vangheluwe, H. 2002. "AToM^3 – A Tool for Multi-formalism and Meta-modelling". In: *European Joint Conference on Theory and Practice of Software (ETAPS), Fundamental Approaches to Software Engineering (FASE),* LNCS 2306, pp. 174-188, Springer-Verlag. Grenoble, France.

Kerckhoffs, E.J.H. and Bobeanu, C.V. 2001. "Systematic Structural Modelling of Discrete Event Systems using Petri Nets". In: Giambiasi, N. and Frydman, C. (Eds.) Proceedings of the 13th European Simulation Symposium "Simulation in Industry", ESS'01, pp. 949-958, Marseille, France, Oct. 18-20, SCS International Publ.

Simchi-Levi, D., Kaminsky, P. and Simchi-Levi, E. 2000. *Designing and Managing the Supply Chain.* McGraw-Hill Higher Education,

Van Landegehem, R. and Bobeanu, C.V. 2002. "Formal Modelling of Supply Chain: An Incremental Approach using Petri Nets". In: Verbraeck, A. and Krug, W. (Eds.). *Proceedings of the 14th European Simulation Symposium "Simulation in Industry",* pp 323-327, Dresden, Oct. 23-26, SCS International Publ.

Vangheluwe, H., Kerckhoffs, E.J.H. and Vansteenkiste, G. 2001. "Computer Automated Modelling of Complex Systems". In: Kerckhoffs, E.J.H. and Snorek, M. (Editors), *Proceedings of the 15th European Simulation Multiconference "Modelling and Simulation 2001", ESM'2001,* pp. 7-18, Prague, Czech Republic, June 6-9, SCS International Publ.

Zeigler, B. P., Praehofer, H. and T. G. Kim. 2000. *Theory of Modeling and Simulation*, 2nd Edition, Academic Press.

PETRI NETS
III

A SIMULATION PLATFORM FOR PETRI NET MODELS OF DYNAMICALLY MODIFIABLE EMBEDDED SYSTEMS

Wang Yan Liu, Carsten Rust, and Friedhelm Stappert
C-LAB, Universität Paderborn
Fürstenallee 11, D-33102 Paderborn,
Germany
Email: {yan|car|fst}@c-lab.de

ABSTRACT

In the paper a simulation platform for dynamically modifiable embedded real-time systems is presented. The tool contributes to a Petri net based design methodology for distributed embedded systems with dynamically modifiable components. Accordingly, the underlying formal model of the platform is a High-Level Petri net model that was extended with constructs for dynamic modification of a net at run-time. Besides the extended Petri net model there is one further characteristic of the presented simulation platform: it supports the integration of a simplified model of the hardware platform on which the considered embedded system shall be implemented.

INTRODUCTION

The design of embedded real-time systems is becoming more and more complicated. The reasons are manifold, e.g. limited resources, reliability requirements, distributed and concurrent behavior and heterogeneity of models as well as target platforms. A further factor raising complexity of embedded systems design is that they tend to be dynamic to an increasing extent. As an example, consider an adaptive robot control where components of the control software are changed at run-time due to results of online learning algorithms. Another application scenario is a group of mobile robots that cooperatively solve a task. Since robots may enter or leave the scenario or just change their location, the entire system is highly dynamic. Also in traditional application domains like automotive systems, dynamically modifying control systems are considered, for instance for the handling of so called fail-over situations, that is in error situations where functionality has to be relocated.

We are currently working towards a methodology for the design of the described systems (Rust et al. 2002). The methodology extends an existing one for distributed embedded systems with a static structure (Rust et al. 2001). The entire methodology is centered around the underlying formal model, a High-Level Petri Net-model (cf. Figure 1). We have chosen a Petri Net model for several reasons, for instance in order to benefit from the multitude of existing verification and analysis methods based on Petri Nets. Petri Nets are used for modeling, simulation and analysis. Synthesis tools are provided for automatically deriving an implementation of a model. Within the overall design, simulation plays an important role. It is an extremely helpful tool when self-modifying systems are considered. The designer needs some support to observe whether his intentions are met by the generation process he designed. Therefore we designed the simulation environment presented in this paper. The environment includes on the one hand execution engines for dynamically modifiable nets. Furthermore classes are provided for building a simplified model of the target hardware. Hence, the platform not only allows for standard Petri Net-simulation, but it is designed for simulating the execution of the implementation of a net on a given target hardware. The realization of the platform is based on object-oriented principles and concepts from Jini (Edwards 1999). The integration of the platform in our methodology can be seen in Figure 1 (*'Simulation of Execution'*).

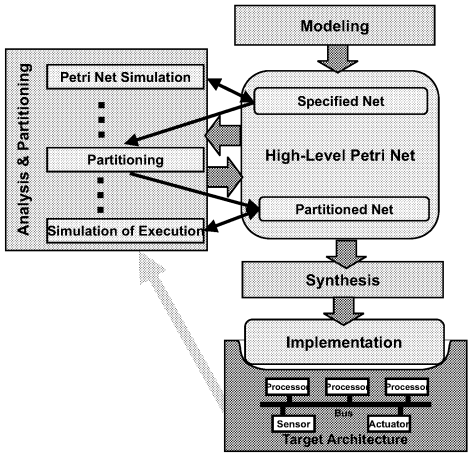


Figure 1. Design Methodology

Many tools for modeling and simulation of Petri nets exist. Examples are Design/CPN for Hierarchical Coloured Petri Nets (www.daimi.au.dk/CPNTools/), Moses for object-oriented Petri nets (http://www.tik.ee.ethz.ch/ moses/), and Renew (www.renew.de) for so called reference nets, another higher-order form of Petri nets. Traditional Petri Net simulation environments, however, are not an adequate tool for our purposes, as they make use of the fixed static structure of ordinary nets. In addition, these environments simulate the behavior of a specified net without respect to the hardware used for implementing the specified system. Hence, the examination of

*This work was supported by the German Science Foundation (DFG) project SFB-376

scenarios where also the hardware is subject to dynamic modification is not possible.

PETRI NET MODEL

In this section, we give a brief overview of the Petri net model supported by our simulation platform. Basically, we use a form of high-level Petri nets. An overview of several high-level Petri net-models is given in (Jensen and Rozenberg 1991). A general introduction into Petri nets can be found for instance in (Murata 1989).

Our hierarchical High-Level Petri Net model is explained with the example net in Figure 2, a small part of a unit for controlling the air conditioning system of a car. Like simple Petri Nets, High-Level nets consist of transitions and places that are connected by directed edges. Places may contain token which can – in contrast to places of simple nets – carry values. According to the annotation of token with values, other components are also annotated, edges for instance with variables that transport token values between places and transitions. The example net contains seven places (e.g. *Event* and *Off*) and eight transitions (T_1, T_2, ...). Transitions have concession to fire when appropriate tokens are available on their input places. Furthermore, in case a condition is defined for the transition, the condition must be fulfilled by the values assigned to the involved tokens. During the firing process transitions may evaluate assignments to variables occuring at their outgoing edges.

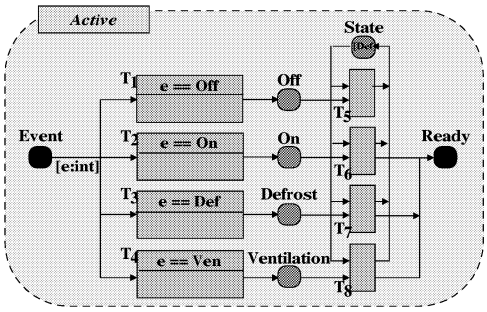


Figure 2. High-Level Petri Net Example

In the example net, the places *Event* and *Ready* are marked as ports so that they can be connected to nodes of other subnets in hierarchical specifications. By putting a token on *Event*, the subnet is activated. Dependent on the value assigned to the token, one of the transitions T_1,..,T_4 fires, which in turn enables one of the transitions T_5,..,T_8. These transitions, which produce a value for the output place *Ready*, are not further specified. In the complete specification, each transition is refined to a subnet that computes – dependent on its input event and the current state buffered on place *State* – the following state. Furthermore, this state is realized in the controlled physical system due to annotations of transitions with calls to according API functions.

Beyond standard constructs of high-level-nets and a hierarchy concept, we added description means for timing to Pr/T–Nets. By this we obtained an adequate modeling means for static distributed embedded systems. However, for modeling systems that are structurally dynamic by nature some further extensions were necessary. In the case of static systems the resulting system can be modelled in advance. In the case of a dynamically reconfigurable one, only the generating system of a set of potentially resulting systems can be provided. Resulting systems are modelled by Extended Pr/T–Nets in our case. An extension of these modeling means is used to describe the generation process as well. The specification of net transformations is based on Petri Net-refinements. Figure 3 shows an example for the refinement of a transition t by a subnet, whose interface consists of transitions only (in the example one input transition and three outputs). The figure also exemplifies the application of the refinement in a net containing transition t: the arcs between the surrounding net and t are replaced by the dashed arcs between the surrounding net and the ports of t.

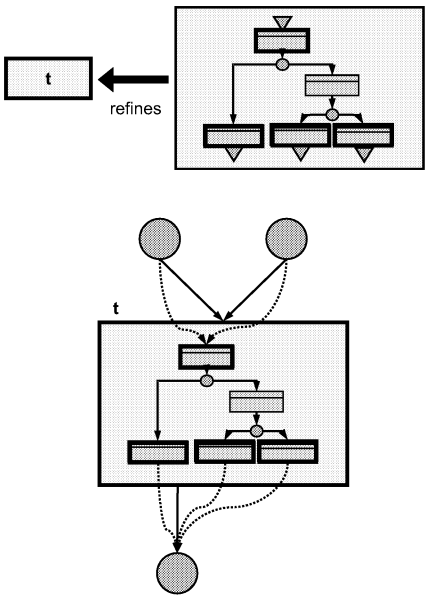


Figure 3. Rule for Dynamic Modifi cation of Refi nements

By annotating a transition with a refinement rule like the one in Figure 3, the specified Petri Net-refinement – usually only used during the specification of a system – is bound to the execution of the transition. Several approaches for Petri Net refinement can be found in literature. For our purposes, we use a set of refinements proposed by Lakos in (Lakos 2000). Besides the above described refinement of a transition by a transition bounded subnet, the set of refinements also includes a place refinement, a type refinement and a net extension. The latter is currently not used in our approach. The refinement of places is similar to that of transitions. Type refinement extends the annotations of a net.

The use of refinements for net modifications combines two advantages. On the one hand, Petri Net refinements are a thoroughly elaborated concept for developing a specification in a well-structured way. They allow for defining transformations that preserve certain relevant properties of the specification. On the other hand they are powerful enough to describe a variety of dynamic modifications. However, in some cases it is necessary to extend the limited set of refinement operations. We therefore offer a second – more generic – approach based on rules as they are used in High-Level replacement systems. For a description of this mechanism we refer to (Rammig and

Rust 2003).
Annotating a transition t_{mod} of a Petri Net N with a refinement rule specifies that by firing of t_{mod}, an arbitrary transition t of N is modified. Usually it is not reasonable to specify rules applicable to arbitrary transitions. Therefore, restrictions on the application can be specified. On the one hand, a scope may be specified that is a subnet in which the transformation may take place. On the other hand, attributes of the component to be modified can be specified in the guard of a transition. That way it is possible for instance to specify precisely to which transition a transformation should be applied by specifying its fully qualified path name. For a definition of the semantics of a transition annotated in such a way, we again refer to (Rammig and Rust 2003).

SIMULATION PLATFORM

In this section, we describe our simulation platform. It is designed to support an engineer in simulating the execution of a Petri net on a model of the target hardware. Application scenarios we are aiming at include dynamic modifications of the application as well as of the hardware it is running on. As mentioned above, the implementation of the platform is mainly based on Jini concepts.
Jini (Edwards 1999) is a programming model mainly based on the Java Remote Method Invocation concept. A Jini application is a federated system of *services* who offer functionality to other members of the system, and *clients* who make use of these services. The key concept is that the actual implementation of a service is completely transparent for a client that wants to use it. A client only receives a *proxy* object of the service from the Jini network. Via this proxy, the client calls methods of the remote service object. Thus, the client doesn't have to care about the actual location and implementation of teh service it uses.
Jini provides the basic software architecture for building and maintaining such a federated system. For example, it provides certain interfaces that have to be implemented in order to build a Jini client or service that can be a part of the system and communicate with the other members. One central entity in a Jini system is the *lookup service*, which registers all services that are currently part of the federation and makes them available to other entities. Furthermore, Jini provides the implementation of several communication protocols: *Lookup* for finding a certain service in the network, *Discovery* to locate a lookup service in the system and *Join* to register a service within the network and make it available to other members. For more details, we refer to the Jini community website (Edwards 1999).
The architecture of the execution platform is exemplified in Figure 4. The platform includes models of all relevant hardware elements. It provides for instance classes for representing sensors and actuators of the physical system. They contain all relevant data for these objects: a name denoting the object, an identifier used to address the object via a communication media (in the example a bus), and some physical properties like the sample rate of a sensor, the type of the sensed value, or the type of an actuator value. Furthermore, data is stored for simulating the communication of these objects with other elements of the platform.

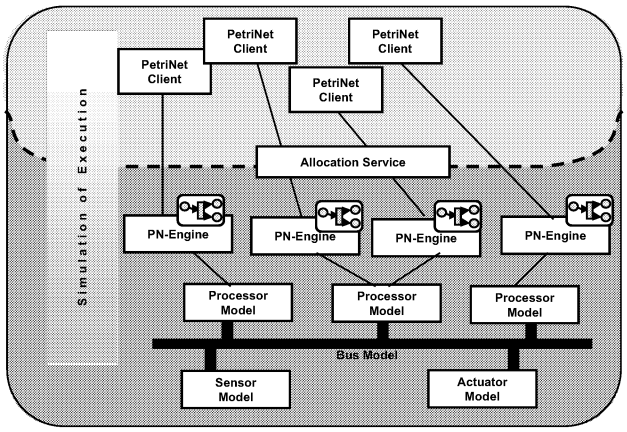


Figure 4. Simulation Platform

For computing devices, a *Processor* class is provided. This class can be subclassed to model different types of processors or microcontrollers. Several units of the application can be executed on each device. On the platform, each unit is a Petri net, in the final implementation it is the realization of a Petri net, typically in C-code. The typical flow of Petri net-execution is the following: a sensor value is sent to a processor or to several processors. The value is forwarded to Petri net-components that are responsible for its processing. The value is put on an input place of each Petri net-component. During the next execution step of the Petri net, which may either be triggered by the value itself or by a timer, the value is processed. In a similar way, values of tokens that appear on output places of Petri net-components are forwarded to actuator objects.
Similar to sensor- and actuator-objects, a processor stores a unique id and further data needed for simulating the communication with other elements. Another important property of a processor object is its memory capacity (c.f. the explanations below). The capacity is specified using simple models, i.e. the largest-possible number of places and transitions contained in the Petri nets currently running on a processor. Different capacity needs for transitions and places respectively can be regarded by means of weights. Further attributes of a processor object are its computation speed, its type, etc. The objects for sensors, actuators and processors build a set of Jini services which are utilized by the Petri Net engines described below.
Communication media can also be represented in the platform by corresponding objects, in the example a bus. The class for these objects provides basic methods for different forms of communication between sensor-, actuator- and processor-objects, for example broadcasting, point-to-point communication, synchronous and asynchronous communication etc. This very general class can be subclassed in order to realize models of specific communication media as for instance a CAN-bus.
The application under development is represented in the platform as a set of Petri nets, i.e. the Petri net-specification which was developed by an engineer, can be used within our platform more ore less directly. We assume that before using the platform, the original net specification was partitioned into single

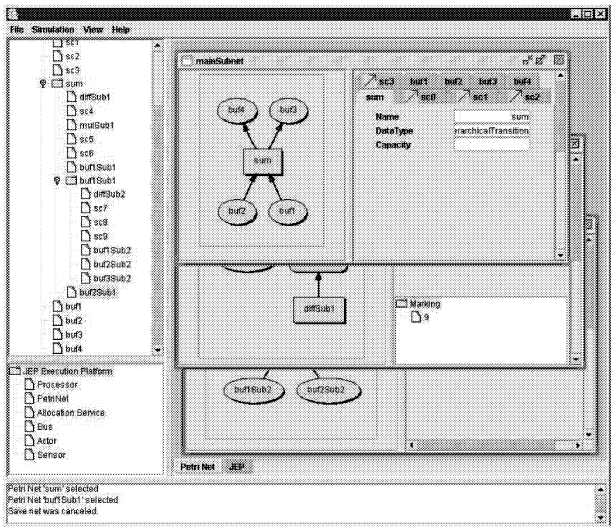


Figure 5. User Interface

units (c.f. Figure 1). The units communicate with each other via shared places. For building partitions, we hence assume an algorithm that – like our partitioning algorithm introduced in (Tacken et al. 1999) – cuts a given net at places, leaving a copy of a cutted place in both partitions built through cutting.

Each component of the partitioned net is handled by an object called *Petri Net Client* (see Figure 4). Properties of these objects are – besides the managed net – the memory needed for execution of the net, the interface of the net, and its connection to other nets. Due to the model used for memory capacities (c.f. the description of processor objects above), the capacity values can be derived from net Petri net-specification. The interface of the net and the connections of input- and output-places to other subnets are also derived from the net specification and its partitioning respectively.

For executing a Petri net on a processor, an *Engine* object is needed (see Figure 4). Engines are the abstraction for the software needed in order to execute code derived from a Petri net-specification. The software represented by an engine typically consists of real-time operating system services as well as of a software library with basic routines for Petri net-execution. On each processor, several engines can be executed, whereby only one engine can be active at each point of time. Opposed to that, the relationship between Petri net clients and engines is unambiguous. The possible number of engines on a processor depends on its capacity, the capacities needed for the associated Petri nets and the capacity needed for the engines, whereby the capacity for one engine is assumed to be constant. An engine inherits the information about the interface of its associated Petri net as well as the information about connections to other nets. Hence, it can process the communication between its Petri net and other components. Considering the Jini-based realization of the platform, engines act as clients of the processor objects, but also as services for execution of Petri nets.

Initially, each Petri net which is part of the application is assigned to an element of the target hardware (model) via an engine. However, when the application is subject to dynamic modifications at run-time an *Allocation Service* is needed, for instance in order to assign a new Petri net to a processor object. The platform contains a default allocation service implemented in Java. It is an extension of the standard Jini lookup service. In order to find an appropriate processor for a not yet allocated Petri net, it stores state information of all processors in the system, compares their free capacity with that needed for the Petri net, and finally chooses one. The allocation service acts as a Jini-Service on the simulation platform that is utilized by Petri net clients.

Figure 5 shows the user interface of our platform. Using commands from the menu bar in the top, Petri nets generated with other tools can be loaded and processed by functions of our platform. The hierarchical structure of the Petri nets is reflected by the tree browser on the left. In the pane on the right, windows showing chosen subnets together with their properties are displayed. One functionality currently supported is of course simulation, either a standard Petri net-simulation or the simulation of an execution on a hardware model. Another functionality is code generation. We currently support the generation of C-Code, SystemC, and Java code. C-Code is generated for software implemenations of system parts on microcontrollers, while the generation of SystemC aims at hardware implementations and interface modelling for Hardware/Software codesign (Rust et al. 2003). The Java code currently generated by our tool is only suitable for prototypes of the system under construction. For import and export of Petri nets, our tool supports XML.

SIMULATION SCENARIOS

Having built up an architecture like the one depicted in Figure 4, it is possible to simulate an execution of the specified application on the specified hardware model. It is possible to consider several scenarios that may occur at run-time, for instance that a processor is removed for some reason or a new processor registers to the system.

A scenario that is realized easily using standard JINI concepts is that of a new processor registering to the entire application. This situation may for instance occur in a robotics application where a group of robots – whose size may change – cooperatively solves a task. Arriving in the application, the processor has to register at the allocation service. Since the allocation service extends the standard lookup service, this procedure is implemented using the *Discovery* and *Join* protocols of Jini.

In the following we will consider examplarily the scenario of a processor leaving the application as depicted in Figure 6. Before the processor (*Old Processor*) leaves the system, it first has to hand over the Petri nets executing on its engines (*Old Engine* in the example) to other processors (*New Processor*). Hence, the scenario starts with *Old Processor* sending a corresponding event to all its engines. Each engine receiving this event sends a message (*1. Query (NewProcessor)* in the Figure) to the allocation service in order to request a new processor for the execution of its Petri net. The relevant data for the Petri net, e.g. its capacity and the Petri net-class, are parameters of the request. Based on these data as well as on the information about availabe processors stored in its internal data structures, the allocation service determines an appro-

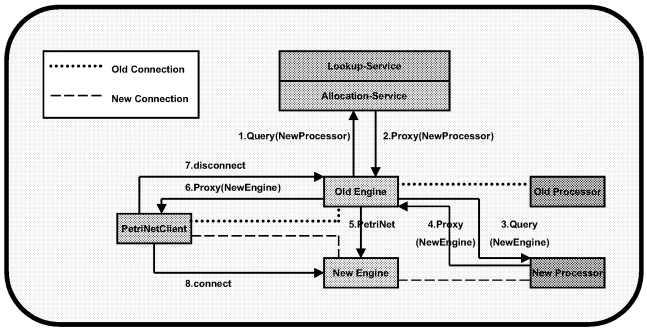


Figure 6. Removal of a processor

priate processor for the net. The old engine receives a proxy object of the new processor as return value of its request (*2. Proxy (NewProcessor)*). Via this proxy object, the old engine initiates the creation of a new engine on the new processor (*3. Query (NewEngine)*). Again, the net's capacity and other relevant data are passed over. According to these data, the new processor creates an engine object. By means of the JINI join protocol, the new engine is registered with the allocation service. The proxy object yielded by the allocation service is returned to the old engine (*4.*). Now the old engine hands over its Petri net to the new engine (*5.*), together with information about which other nets the input- and output-ports of this net are connected with. Hence, the moved Petri net is easily reconnected with the overall net structure. After this, the old engine sends the proxy of the new engine to its Petri Net Client (*6.*), thereby enabling the client to execute the Petri net via the new engine (*7. and 8.*). The old engine forwards all incoming requests to the new engine until the Petri Net Client has acknowledged the receipt of the new proxy. After receiving the acknowledgement from its Petri Net Client the old engine logs off from its processor and terminates. When all its engines are terminated, the old processor itself logs off from the allocation service and thereby leaves the entire application.

APPLICATION EXAMPLE

We are currently evualating our approach with an application based on Khepera minirobots (K-Team 2002). Such a robot includes a PowerPC microcontroller, 8 sensors and two motors

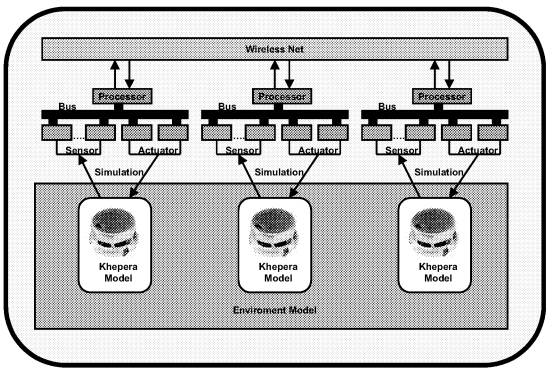


Figure 7. Application Example

for the left and the right wheel. The robots communicate via radio. Currently we have implemented relatively simple scenarios where each robot has its own defined behavior, e.g. avoiding obstacles or following another robot. The goal of our experiments is to implement a distributed net of Khepera robots that cooperatively solve a certain task, like e.g. collecting items distributed in a field and transporting them to a designated area. The behavior of each robot was specified as a Petri net. From the Petri net models C-code is generated, compiled and executed on the Khepera's microcontroller. The mapping of the Khepera's hardware components to corresponding elements of our simulation platform is shown in Figure 7.

CONCLUSION

We have presented a simulation platform for dynamically modifiable embedded real-time systems. The tool contributes to a Petri net based design methodology for distributed embedded systems with dynamically modifiable components. Besides the simulation of extended Petri net models, the platform also supports a simplified model of all relevant hardware elements on which the system under design shall be implemented. Furthermore, an application scenario with a group of Khepera minirobots was described.

REFERENCES

Edwards, W. K. 1999. *Core Jini*, The Sun Microsystem Press Java Series, Prentice Hall PTR.

Jensen, K. and Rozenberg, G. 1991. *High-Level Petri Nets*, Springer Verlag.

K-Team. 2002. 'Mobile Robots as Research Tools: The K-Robot Family', http://k-team.com/.

Lakos, C. 2000. Composing abstractions of coloured petri nets., *in* Nielsen, M. and Simpson, D., eds, 'Lecture Notes in Computer Science: 21st International Conference on Application and Theory of Petri Nets (ICATPN 2000), Aarhus, Denmark, June 2000', Vol. 1825, Springer-Verlag, pp. 323–345.

Murata, T. 1989 Petri Nets: Properties, Analysis and Applications., *in* 'Proceedings of the IEEE', pp. 541–580. NewsletterInfo: 33Published as Proceedings of the IEEE, volume 77, number 4.

Rammig, F. and Rust, C. 2003. Modeling of Dynamically Modifi - able Embedded Real-Time Systems. *9-th IEEE International Workshop on Object-oriented Real-time Dependable Systems* (WORDS 2003F).

Rust, C., Rettberg, A. and Gossens, K. 2003. From High-Level Petri Nets to SystemC. *IEEE Conference on Systems, Man, and Cybernetics*, (SMC'03).

Rust, C., Stappert, F. and Bernhardi-Grisson, R. 2002. Petri Net Based Design of Reconfi gurable Embedded Real-Time Systems, *in* 'Distributed And Parallel Embedded Systems', Kluwer Academic Publishers, pp. 41–50.

Rust, C., Tacken, J. and Böke, C. 2001. Pr/T-Net Based Seamless Design of Embedded Real-Time Systems, *in* J.-M. Colom and M. Kounty, eds, 'LNCS 2075; International Conference in Application and Theory of Petri Nets (ICATPN)', Vol. 2075, Springer Verlag, Newcastle upon Tyne, U. K., pp. 343–362.

Tacken, J., Rust, C. and B.Kleinjohann. 1999. A method for prepartitioning of petri net models for parallel embedded real-time systems, *in* 'Proc. of the 6th Annual Australasian Conference on Parallel And Real-Time Systems (PART '99)', Melbourne, Australia.

INTEGRATING LOAD BALANCING INTO PETRI-NET BASED EMBEDDED SYSTEM DESIGN *

Carsten Rust and Friedhelm Stappert
C-LAB, Universität Paderborn
Fürstenallee 11, D-33102 Paderborn
{fst|car}@c-lab.de

Stefan Schamberger
Universität Paderborn
Fürstenallee 11, D-33102 Paderborn
schaum@uni-paderborn.de

ABSTRACT

This paper introduces a Petri net model of a load balancing component for dynamically modifiable embedded real-time systems. The presented work contributes to the extension of an existing Petri net based design methodology towards the handling of dynamically modifiable systems. Modeling the load balancing algorithm in the same formal model as the other components of the system under construction enables the engineer to simulate and analyze the overall system. This paper presents our load balancing model, gives a brief overview on diffusive load balancing algorithms and describes their integration into the application. Furthermore, some first results with our approach are presented.

INTRODUCTION

Embedded systems have evolved from small systems realized with very limited hardware devices to complex heterogeneous systems realized on a set of interconnected computing devices. Furthermore, there is a trend towards an increasing degree of dynamics in embedded systems. In several application areas, scenarios including dynamically modifiable components are discussed. As an example, consider an adaptive robot control, where components of the control software are changed at run-time due to results of online learning algorithms. Also in traditional application domains like automotive systems, scenarios are discussed where software components are replaced by other components during run-time, for instance within the infotainment system. Another typical example for a system with dynamically changing structure is a mobile ad hoc network (MANET). Possible examples for hosts building a MANET are mobile phones, PDA's, or robots that cooperatively solve a task.

An essential step in the design of embedded systems is the allocation of tasks to computing devices. Usually, the allocation is computed statically in advance. For dynamic systems however, an online component for load balancing has to be added, which is responsible for modifying the intial allocation if necessary.

*This work was supported by the German Science Foundation (DFG) project SFB-376

In this paper we present our Petri net realization of a load balancing algorithm for dynamically modifiable embedded systems. The presented approach is integrated into an existing design methodology for distributed embedded real-time systems including dynamically evolving components. The methodology covers the whole design flow, reaching from modeling of a system on an abstract level via analysis and partitioning down to the synthesis of an implementation on a given target architecture. The main characteristic of the methodology is that during analysis and synthesis the whole system under construction is present in one uniform formal model, a High-Level Petri net model. This enables evaluation and analysis of all components including their interaction. When it comes to the realization of a dynamic system, we have to take into account that load balancing – opposed to static design steps like scheduling and allocation – is a part of the implementation. Hence, it should be subject to analysis methods like the other parts. In order to integrate load balancing into the overall design, we have to build a model in our Petri Net formalism, which is the topic of this paper.

Many design methodologies and modeling means for embedded real-time systems exist, based on a variety of formal models. Several methodologies originate from the area of hardware/software-codesign, e. g. Ptolemy (Lee et al. 2001) and SPI (Jersak et al. 2000). These systems usually provide only limited support for dynamic systems. Ptolemy for instance allows for creating tasks at run-time, but they cannot properly be connected to existing tasks. Consequently, other design approaches do – to our knowledge – not consider the integration of load balancing into the overall system design as proposed in this paper.

In the remaining sections of the paper, we first provide some background concerning the existing design methodology (Section: Design Methodology) and the applied load balancing strategy (Section: Diffusive Load Balancing). Our Petri Net-model for load balancing and its integration into an application is presented in Section "Petri-Net Specification". In Section "Experiments", first results with our approach are described.

DESIGN METHODOLOGY

In this section we give a brief overview of the methodology to which the presented approach contributes and the underlying Petri netformalism. The methodology proposes the design flow depicted in Figure 1. It is divided into the three stages *Modeling*, *Analysis and Partitioning*, and *Synthesis*. Within the stage of modeling, a heterogeneous model of the system under construction – specified using languages from different application domains – is transformed into one unique high-level Petri net. In the second stage, Petri net analysis and timing analysis methods are applied in order to validate functional as well as temporal requirements and furthermore in order to gather information for an effective implementation of the system. The implementation is generated in the final stage of synthesis.

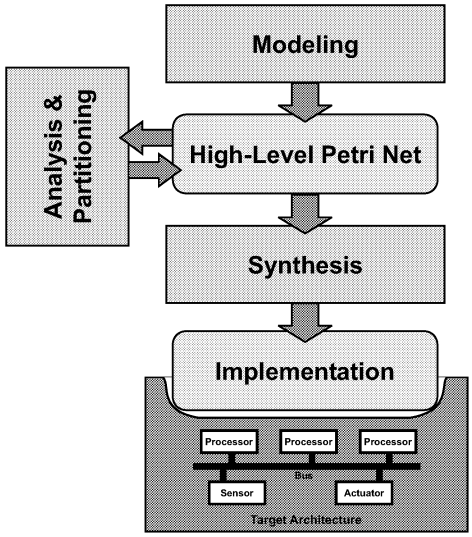


Figure 1. Design Flow

A more detailed overview of the methodology for the design of static systems without dynmically changing components is for instance given in (Rust et al. 2001). In (Rust et al. 2002) ideas for the extension to dynamic systems are presented. This extension has an impact on all stages of the design flow. First, the underlying Petri Netformalism – which is described below – has to be extended with constructs for self modification in order to obtain an adequate modeling means. For the enhanced model, new tools for simulation, analysis and synthesis are needed. As to synthesis the main task is to extend existing tools by run-time services for the management of dynamic reconfigurations. One of these services is the component for load balancing described in this paper.

The underlying Petri net formalism of the methodology is a high-level model. An overview of several high-level Petri net models is given in (Jensen and Rozenberg 1991). A general introduction into Petri nets can be found for instance in (Murata 1989). Petri nets are bipartite directed graphs augmented by a marking. The Petri net graph consists of a finite set of places, a finite set of transitions, and directed edges from places to transitions and from transitions to places respectively. Places model conditions. For this purpose they may be marked by tokens. Driven by specific firing rules, a transition can fire based on the local marking of those places it is directly connected with. By firing, the marking of these places is modified. In the case of high-level nets the tokens are typed individuals. The other net components are annotated accordingly: places with data types, edges with variable expressions and transitions with a guard and a set of variable assignments. Now a transition can fire only if the formal edge expressions can be unified with actually available tokens and this unification passes the guard expression of the transition. By firing the input tokens are consumed and calculations associated with the transition are executed. That way new tokens are produced and routed to output places according to variable expressions annotating the output edges of the transition. An example for a high-level Petri net is the net in Figure 2, our specification of the load balancing component. It will be explained in Section .

Beyond standard constructs of high-level Petri nets, our formal model includes a hierarchy concept in order to support easy modeling of complex systems as well as description means for timing to Pr/T–Nets. Furthermore, we added constructs for specifying dynamic modifications of the model. The resulting self-modifying net model allows to annotate transitions with rules for net transformations. Firing a suchlike annotated transition leads to a corresponding modification in the current net. The modification can be an instanciation of a subnet, but also a structural modification of an existing subnet. For more details we refer to (Rammig and Rust 2003).

DIFFUSIVE LOAD BALANCING

To distribute the computational load, we apply diffusive load balancing schemes. These schemes work in iterations and require communication with adjacent nodes only. This is important since no additional routing mechanisms are needed. Furthermore, it has been shown that these schemes always compute the l_2-minimal flow ensuring a small number of migrations and also work in inhomogenous and asynchronous networks. Due to the space restrictions we will only give a brief overview to this topic and refer the reader to (Elsässer et al. 2002) and the references therein.

Let $G = (V, E)$ be the connected, undirected graph representing the network topology. We denote the nodes' computing power with $p_v \in \mathbb{R}, v \in V$ and the links' communication costs with $c_e \in \mathbb{R}, e \in E$. Let $w_v \in \mathbb{R}$ be the work load of node $v \in V$ and $\overline{w} := \sum w_v / \sum p_v \cdot (p_1, \ldots, p_{|V|})$ the vector of the propotional balanced load. The task is now to compute a balancing flow $x \in \mathbb{R}^{|E|}$, such that $Ax = w - \overline{w}$ with $A \in \{-1, 0, +1\} \in \mathbb{R}^{|V| \times |E|}$ defined as the node edge incidence matrix of G.

The simplest diffusive method also called First–Order–Scheme

(FOS) performs on each node $v_i \in V$ the iteration:

$$\forall e = (v_i, v_j) \in E: \quad y_e^{k-1} = \frac{\alpha_e}{c_e}\left(\frac{w_i^{k-1}}{p_i} - \frac{w_j^{k-1}}{p_j}\right)$$
$$x_e^k = x_e^{k-1} + y_e^{k-1} \qquad (1)$$
$$\text{and} \quad w_i^k = w_i^{k-1} - \sum_{e=(v_i,v_j)\in E} y_e^{k-1}$$

where y_e^k is the amount of load sent via edge e in iteration k and the α_e are properly chosen parameters, e.g. $\alpha_{(v_i,v_j)} = 1/(1 + deg(v_i))$. In matrix notation, this can be written as $w^k = Mw^{k-1}$ with the diffusion matrix $M = I - \alpha\tilde{L}P^{-1} \in \mathbb{R}^{|V|\times|V|}$. Here, $L = \tilde{A}\tilde{A}^T$ is the generalized Laplacian, meaning $\tilde{A} = AC^{-1}$ and $C = diag(\sqrt{c_e}) \in \mathbb{R}^{|E|\times|E|}$, and $P = diag(p_v) \in \mathbb{R}^{|V|\times|V|}$.

A quadratic increase in the convergence rate can be achived applying the Second-Order-Scheme (SOS). This scheme takes the work load of the previous iteration into account. In matrix notation, it can be written as $w^1 = Mw^0$; $w^k = \beta Mw^{k-1} + (1 - \beta)w^{k-2}$. It converges for $\beta \in (0,2)$ and needs the least number of iterations for $\beta = 2/\left(1 + \sqrt{1 - (\frac{\lambda_m - \lambda_2}{\lambda_m + \lambda_2})^2}\right)$, where $0 = \lambda_1 < \lambda_2 < \ldots < \lambda_m$ are the m distinct eigenvalues of $\tilde{L}C^{-1}$.

The knowledge of all distinct non-zero eigenvalues λ_k of $\tilde{L}C^{-1}$ is needed for the Optimal-Scheme (OPT). Replacing α_k by $1/\lambda_k$ in equation (1), only $m - 1$ iterations are needed to fully balance the load. If the topology is fixed (e.g. some hardware), one can precompute the eigenvalues and store them on the nodes. Furthermore, it is also possible to look for topologies with a small number of distinct eigenvalues and thus allowing efficient load balancing.

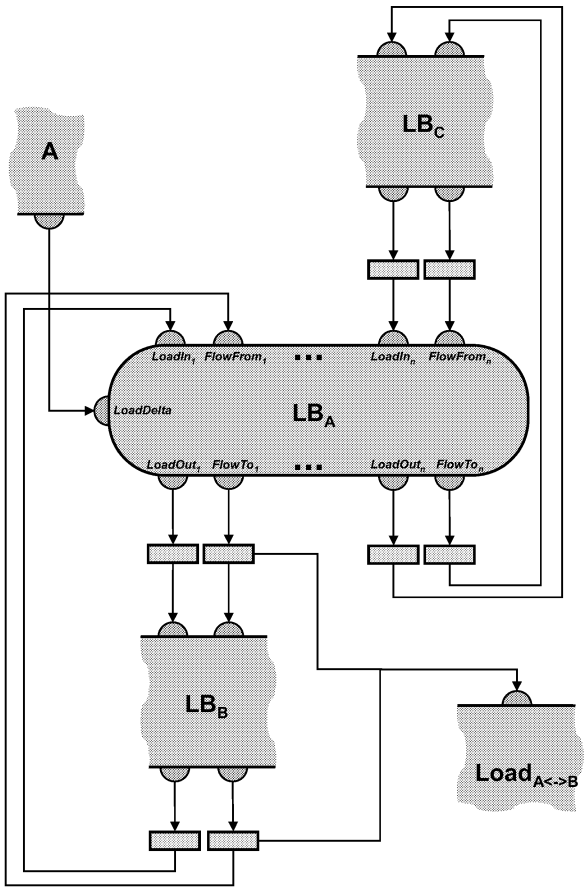


Figure 2. Top-Level Petri Net

PETRI-NET SPECIFICATION

This section presents our Petri net implementation of the FOS scheme described in the previous section. In the application scenarios we are aiming at, it is not necessary to get equal load on all nodes. Instead, it is sufficient to reach a load situation where each node can perform all its tasks fast enough, meeting all corresponding real-time constraints. The iterative process of load distribution is started when the load of one or more nodes has become inacceptably high, i.e. its load exceeds a certain threshold. The process is stopped when no node is loaded too heavily, i.e. its load does not exceed the threshold. The load of a given node may change either during load balancing when it receives load from its neighbors, or internally due to dynamic modifications in the subsystem realized on the respective node.

The FOS calculation proposed in this paper is implemented completely asynchronous. Each node decides on its own whether a load distribution should be initiated. If it does so, it reads the current load situation from all its neighbors and calculates the flow that should be sent to each of them. This, in turn, may trigger the distribution process in some of the neighbor nodes. This process continues until all involved nodes have reduced their load below the given threshold. Thus, no central instance is needed to coordinate the activities of the nodes.

In Figure 2 the top-level view of our Petri net model is depicted exemplarily. It shows a load balancing component LB_A of a Petri net A. The net is connected to its immediate neighbors LB_B and LB_C by means of the input-ports $LoadIn_i$ and $FlowIn_i$ from which it reads their load and flow, i.e. the amount of load the corresponding neighbor has sent. The net's own values for load and flow are sent to the corresponding neighbors via the output-ports $LoadOut_i$ and $FlowTo_i$. When the load of net A changes internally, the load-balancing component LB_A is informed via the input-port $LoadDelta$. Furthermore, for each neighbor N of A the component $Load_{AN}$ (depicted for neighbor B only in Figure 2) is informed whenever load has to be moved from one node to the other. While LB_A only computes the values for the load change, $Load_{AN}$ is responsible for the actual movement of computational load, i.e. it decides which processes shall migrate from one node to the other. The movement of computational load is specified by Petri net transformations as they were described in Section . Their realization in the implementation is beyond the scope of this paper and not further explained here. But without going into the details of the implementation it is obvious that not every arbitrary flow

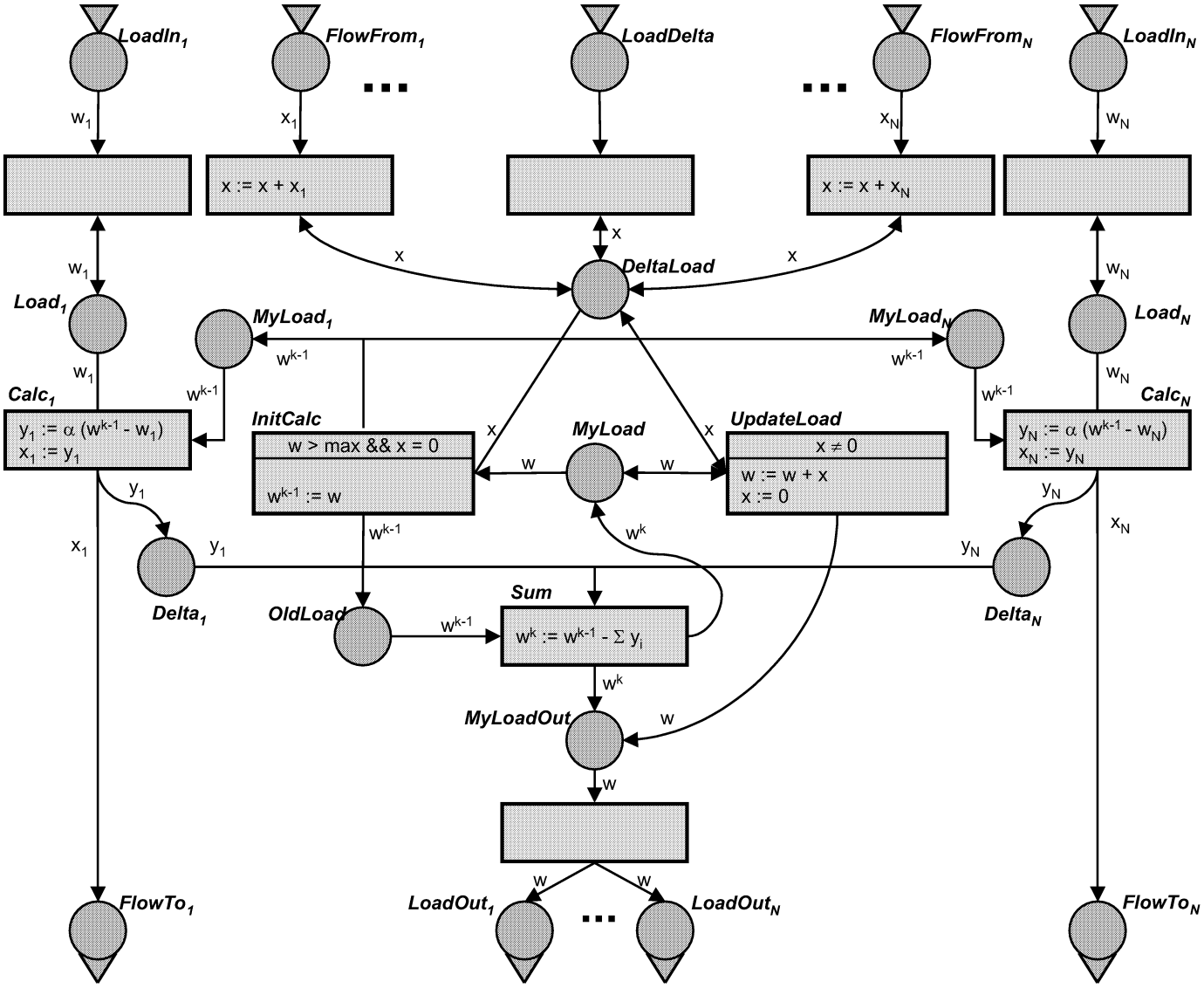


Figure 3. FOS Petri Net

value calculated by the diffusive schemes described above can be realized. Instead, the net transformation is chosen, which realizes the greatest flow smaller than the calculated value. We always realize flows less or equal than the optimal value, since the choice of a greater flow often leads to circular load balancing operations. The difference between a calculated flow value and the realized one is sent to the load balancing component via the *LoadDelta* port.

Figure 3 shows the Petri net model specifying the calculations performed locally at each node. The variable names in the annotations of the edges and transitions correspond to the nomenclature used in Section . For simplicity reasons, we assume that the computing power p_i for each node i and the communication cost c_e for each edge e are equal. Therefore, the corresponding factors are left out in the figure. As suggested in Section , α is set to $1/(1+deg(i))$ for each node i.

The current load situation of the node is stored in the place *MyLoad* and is also distributed to the neighbor nodes via the output-places $LoadOut_i$. If the net receives load from its neighbors via the input-places $FlowFrom_i$ or the load changes internally via the input-place *LoadDelta*, *MyLoad* is updated by the transition $UpdateLoad$.

A new iteration of FOS calculation is triggered by the transition *InitCalc* when it realizes that the current load is too high (condition '$w > max$'). *InitCalc* places a token with the current load value on each place $MyLoad_i$, thereby enabling the corresponding transitions $Calc_i$. Note that there is one place $MyLoad_i$ and one transition $Calc_i$ for each neighbor i of the given net. The transitions $Calc_i$ then compute the amount of load that should be sent to the corresponding neighbor i and store it in the place $FlowTo_i$. Furthermore, the computed values are collected and subtracted from the current load by the

transition *Sum* which updates the current load value in place *MyLoad*.

EXPERIMENTS

Currently we are evaluating our load balancing method by means of random graphs. Using a library with small Petri net

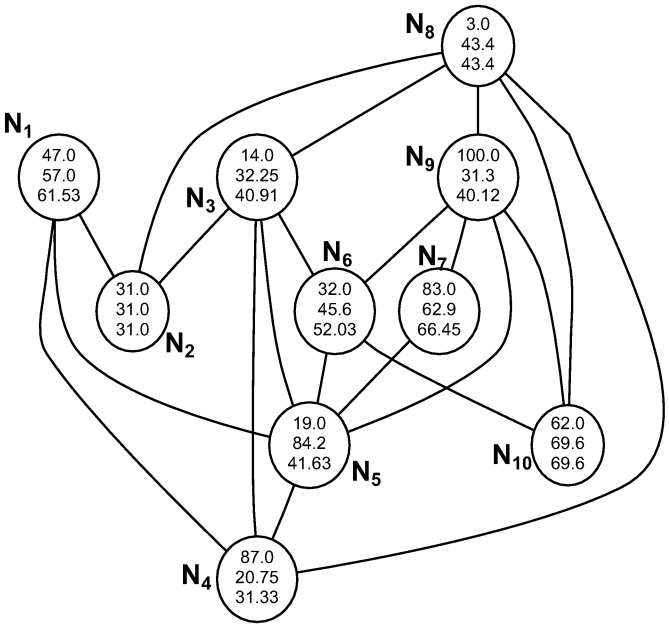


Figure 4. Example Graph

components, a Petri net of arbitrary size is built up automatically. The generated net is organized into units whose load is estimated by the number of their transitions. The Petri nets are self-modifying, i. e. the load of each unit changes over time.

Figure 4 shows the top-level view of a generated net with ten units. Each node represents one unit and the edges of the graph represent neighbor relations between the subnets. The values annotating the graph nodes are load values for three successive situations. We assume that the maximal acceptable load value is 80. In the first situation, this value is exceeded for three nodes: N_4, N_7, and N_9. After one iteration of load balancing, i. e. after each of these nodes has sent load to its neighbours once, the second value is obtained. After a second iteration, the last values are reached and the algorithm terminates since a situation is reached where no node exeeceds the maximal load value of 80. Further iterations would lead to a situation where the load is distributed evenly over all nodes.

CONCLUSION AND FUTURE WORK

We presented a Petri net model of a load balancing component for dynamically modifiable embedded real-time systems. The model itself as well as its integration into an application were described. Some first experiments with the load balancing implementation were described. As to the experimental results however, we only have shown so far that the implemented strategy works for generated random nets and leads to sufficiently balanced loads after a small number of iterations.

Future work includes performance investigations with very large applications. Such nets could easily be generated using our random net generation. Also, further investigations with real-world applications are planned, which may have additional properties not covered by the random nets we currently use.

Currently, we can only simulate top-level nets with static structure. For the applications we are aiming at, it is necessary to realize dynamic behavior – i.e. nodes can enter or leave the overall system – on this level, too. For example, when a node leaves the system, its entire load has to be moved to its neighbors at once, and the top-level net structure has to be reconfigured accordingly.

At the moment, we are working on the implementation of the $Load_{AN}$ component shown in Figure 2, i.e. the actual movement of Petri net components from one node to another.

REFERENCES

Elsässer, R., Monien, B. and Preis, R. 2002. 'Diffusion schemes for load balancing on heterogeneous networks', *Theory of Computing Systems* **35**, 305–320.

Lee, E. et al. 2001. Overview of the Ptolemy Project, Technical Memorandum M01/11, Dept. EECS, University of California, Berkeley, CA 94720.

Jensen, K. and Rozenberg, G. 1991. *High-Level Petri Nets*, Springer Verlag.

Jersak, M., Ziegenbein, D., Wolf, F., Richter, K., Ernst, R., Cieslok, F., Teich, J., Strehl, K. and Thiele, L. 2000. Embedded System Design using the SPI Workbench, *in* 'Proc. FDL'00, Forum on Design Languages 2000', Tübingen, Germany.

Murata, T. 1989. 'Petri Nets: Properties, Analysis and Applications', *Proceedings of the IEEE* **77**(4), 541–580.

Rammig, F. and Rust, C. 2003. Modeling of Dynamically Modifiable Embedded Real-Time Systems. 9-th IEEE International Workshop on Object-oriented Real-time Dependable Systems (WORDS 2003F).

Rust, C., Stappert, F. and Bernhardi-Grisson, R. 2002. Petri Net Based Design of Reconfigurable Embedded Real-Time Systems, *in* 'Distributed And Parallel Embedded Systems', Kluwer Academic Publishers, pp. 41–50.

Rust, C., Tacken, J. and Böke, C. 2001. Pr/T-Net Based Seamless Design of Embedded Real-Time Systems, *in* J.-M. Colom and M. Kounty, eds, 'LNCS 2075; International Conference in Application and Theory of Petri Nets (ICATPN)', Vol. 2075, Springer Verlag, Newcastle upon Tyne, U. K., pp. 343–362.

MODELING OF MANUFACTURING SYSTEM FOR PERFORMANCE ANALYSIS: AN APPROACH BASED ON GSPN

Mauro Silva
Paulo Maciel
Wellington Silva
Centro de Informática - Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7.851 - 50.732 - 970 - Recife - PE - Brazil
E-mail:{mjcs, prmm, wjs}@cin.ufpe.br

KEYWORDS
Production System, Performance Analysis, Generalized Stochastic Petri Nets, Modeling and Fault.

ABSTRACT

A production system transforms raw material into products. Such products may be composed of parts in a given determined order. The products' production planning depends on the amount of available capital for investments among other aspects.
Based on the planning, the number and type of resources used on the shop floor have to be defined as well as the analysis of the effects related to time-to-fail of machines on the average production time.

This work proposes an approach based on Generalize Stochastic Petri Nets for modeling manufacturing systems, where such models allow for resource estimation and machines fault influence on system's performance.

1. Introduction

Modern manufacturing systems have to deal with requirements such as batch size, product variety, production demands, cost and quality.

Besides, due to actual world market, efforts have been made to improve competitiveness of established industry.

Petri Nets are a powerful family of formal description techniques able to model a large variety of problems such as concurrent, asynchronous and non-deterministic systems (Murata 1989; Desrochers and Al-Jaar 1995). However, Petri Nets are not only restricted to design modeling. It is also important for analysis of the modeled system. Several approaches have been developed for qualitative and quantitative analysis of Petri Net models. This work proposes a modeling methodology for manufacturing systems aiming the performance analysis considering resource constraints and machines fault.

Therefore, one important issue is to define the minimal number of resources needed to execute the planning in a minimal time, also considering machines unreliability.

Section 2 presents an overview of the production systems. Section 3 introduces basic concepts on Generalize Stochastic Petri Nets (GSPN). Section 4 discusses the proposed modeling approach. Section 5 presents some estimates obtained, considering a model for a given case study. Section 6 concludes the work and introduces some further objectives.

2. Production System

A production system transforms raw material into products, where products may be composed of *parts* in a given determined order. Therefore, there is a flow of raw material or parts inside the system (see Figure 1). The production planning depends on economics factors and also the amount of capital. These factors are basically related to market demands, customer preferences, availability of raw material and supplies. In order to execute the planning, *services orders* are delivered to the system. A service order is a no empty finite set of *service requests* specifying how to obtain products. Therefore, a process (or a job) is a service order in execution. The total time required to complete all the process is denoted by *makespam time* (τ_m). Production Systems have two structures that may be described separately: the *control* and the *plant*. The control manages the concurrent execution of service orders delivered to the system and informs the production management about the service

executed (Murata 1989). The plant is composed by a number of equipment having resources used to perform different functions. Therefore, pieces of equipment are controlled objects of the plant.

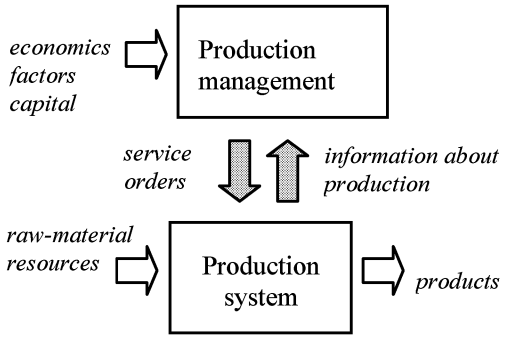


Figure 1: Overview of Production Systems

The plant of the Discrete Production Systems is basically composed by production (components that transform raw material into parts or final products), storage (denoted as buffers, storage of raw material, parts or products) and transportation resources (used to transport raw material, parts or products between buffers of the system) (Lee and DiCesare 1987). The machine representation is shown in Figure 2 as follows: the two rectangles inside the square are used to show fail rate and repair rate; the central circle is used to represent available machine production resources; the two half circles are used to represent input and output buffers. In order to represent machines, the values of the fail and repair rates assume an exponentially distributed firing time.

3. Generalized Stochastic Petri Nets

In 1984, Ajmone Marsan, Balbo, and Conte introduced *generalized stochastic Petri net* (GSPN), in which transitions have either an exponentially distributed firing delay or fire without delay (Marsan et al. 1991). They showed that the stochastic process underlying a GSPN could be represented as a continuous-time Markov chain. Thus, GSPN belong to the class of Markovian modeling formalisms and their stationary analysis requires the solution of a linear system of equations. The steady-state probabilities obtained from the Markov chain are used to compute the expected number of tokens in a place or set of places, the probability that a place is not empty, and the probability that a transition is enabled. Performance measures such as average production rate, average in-process inventory, and average resource utilization can also be computed from the steady-state probabilities.

In the following, a formal definition of GSPN is presented.

GSPN – *Let GSPN = (P, T, I, O, H, Π, M_0, D, W), where:*

- ***P** is a finite set of places. Each place may contain a finite number of tokens. A marking $M_k \in N^{|P|}$ is a vector of nonnegative integers and defines the number of tokens in each place $p_i \in P$. For a marking M_k, the number of tokens in a particular place p_i is given by the i-th component of M_k. The number of tokens of a particular place p_i can also be indicated by $\#p_i$.*

- ***T is** a finite set of transitions. Of course, the set of transitions is disjoint with the set of places, i.e., $P \cap T = \phi$. In a GSPN, the set of transitions can be partitioned into two disjointed subsets, comprising immediate and exponential transition, respectively. These subsets are denoted by T_{im} and T_{exp} with $T = T_{im} \cup Texp$.*

- ***I**, **O** and **H** denote the input, output and inhibitor functions which map transitions on multisets of places. The multiplicities of the input arc from a place p_i to a transition t_j, the output arc from t_j to p_i, and the inhibitor arc from p_i to t_j are denoted by $i_i(t_j)$, $o_i(t_j)$, and $h_i(t_j)$, respectively. Formally: $\forall p_i \in P$, $\forall t_j \in T$:*
  - $i_i(t_j) : P \times T \rightarrow N$
  - $o_i(t_j) : T \times P \rightarrow N$
  - $h_i(t_j) : P \times T \rightarrow N$

- ***Π** is the firing priority function for immediate transitions which specifies the firing priority level for all immediate transition. Formally: $\forall t_j \in T_{im} : \Pi(t_j) : T_{im} \rightarrow N$*

- ***M_0** is the initial marking of the GSPN, $M_0 \in N^{|P|}$.*

- ***D** is the firing delay function for timed transitions of the GSPN. It specifies the mean firing delay for each exponential transition. Formally: $\forall t_j \in T_{exp} : D(t_j) : T_{exp} \rightarrow (0, \infty)$.*

- ***W** is the firing weight function, which specifies the firing weight associated with each immediate transition. We denote the firing weight of immediate transition t_j by w_j. Formally: $\forall t_j \in T_{im} : W(t_j) : T_{im} \rightarrow (0, \infty)$.*

4. Performance Modeling

This section presents the proposed modeling method. First, however, a small example is introduced. This example is used for explaining the proposed modeling methodology. A discrete production system is presented in Figure 2. This system has two manufacturing cells, denoted $Cell_1$ and $Cell_2$. The robot R_2 loads the two input buffers of $Cell_2$ from output buffer of $Cell_1$. The robot R1 loads machine M3 from machines M1 or M2 and the robot R3 loads machine M6 from machines M4 or M5.

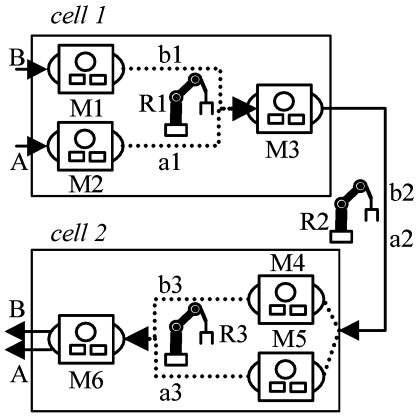


Figure 2: Discrete Production System

As mentioned in Section 2, a process is a service order in execution. A process is executed in phases where, in each phase, one or more resources are allocated. For the system showed in Figure 2, the phases are:

Phases of Processes A (product A):

- FA1: raw material of product A is processed in machine M2 manufacturing the sub-product (a1); FA2: the robot R1 transports the sub-product (a1) from machine M2 to the machine M3; FA3: the sub-product (a1) is processed in machine M3 producing the sub-product (a2); FA4: the robot R2 transports the sub-product (a1) from machine M3 to the machine M5; FA5: the sub-product (a2) is processed in machine M5 manufacturing the sub-product (a3); FA6: the robot R3 transports the sub-product (a3) from machine M5 to the machine M6; FA7: the sub-product (a3) is processed in machine M6 producing the product A.

Phases of Processes B (product B):

- FB1: raw material of product B is processed in machine M1 manufacturing the sub-product (b1); FB2: the robot R1 transports the sub-product (b1) from machine M1 to the machine M3; FB3: the sub-product (b1) is processed in machine M3 producing the sub-product (b2); FB4: the robot R2 transports the sub-product (b1) from machine M3 to the machine M4; FB5: the sub-product (b2) is processed in machine M4 manufacturing the sub-product (b3); FB6: the robot R3 transports the sub-product (b3) from machine M4 to the machine M6; FB7: the sub-product (b3) is processed in machine M6 producing the product B.

Table 1: Phase's

			Processing Time	Time-to-fail	Time-to-repair
Process	A	FA1	1	1000	10
		FA2	1	1000	20
		FA3	1	1000	10
		FA4	1	1000	10
		FA5	1	1000	10
		FA6	1	1000	20
		FA7	1	1000	10
	B	FB1	1	2000	20
		FB2	1	1000	10
		FB3	1	1000	10
		FB4	1	1000	10
		FB5	1	2000	20
		FB6	1	1000	20
		FB7	1	1000	10

One should observe that, the end of a phase is the beginning of the next phase. For instance, the end of phase FA1 is the beginning of phase FA2. It is worth pointing out that any phase occurs between two buffers of the system. Therefore, a buffer is related to the beginning or to the end of a phase. In this work, a buffer is called *source* if it is related to the beginning of a phase and *target*, if it is related to the end of a phase. The phase may be concurrently executed, that is, the activities of phases may occur in parallel.

In the manufacturing system, there are some constraints that must be satisfied. These constraints may be related to production rate, failure and repair rates, and resources sharing, etc. The main goal of this work is to analyze the effect of these constraints in the makespam

time. The makespam time is the total time required to complete all the jobs. The time-to-fail is the amount of time that the machine has available for work. Another way of saying this is that it is the time between completion of the last repair and the time of the next failure. When a machine fails it takes a certain amount time-to-repair. So the time-to-repair is the time that elapses between the last machine failure and the time when it is available for working again (Desrochers and Al-Jaar 1995).

5. Formal Model

The proposed methodology for modeling production system is based on Generalized Stochastic Petri Net models. These models are: Input-Flow Model (IFM), Output-Flow Model (OFM), Closure Model (CM), Phase Model (PM), and The Process Models (PRM).

Discrete Production System Model – *Let S be a discrete production system. SM = (IFM, OFM, CM, PRM) defines discrete production system model, where IFM is the input flow mode, OFM is the output flow model, CM is the closure model and PRM is the process model.*

The Input-Flow model describes the arrival of service orders. This model is formally defined below.

Input-Flow Model – Let IFM = (P, T, I, O, H, Π, M_0, D, W) be a GSPN defining the input-flow model of service orders. T = $\{t_0\}$, where t_0 is an immediate transition (T_{im}). $p_{in,i} \in P$, where $p_0 \neq p_{in,i}$. p_0 is named start place. A token in place $p_{in,i} \in P$ models a service order. $I(p_0, t_0) = 1$. $O(t_0, p_{in,i}) = n_i$, $\forall\ p_{in,i} \in P$, where n_i is the number of service orders related to the product i. $\Pi(t_0)=W(t_0)=1$.

The Output-Flow model represents the conclusion of all processes. The formal definition of this model is given in the following:

Output-Flow Model – *Let OFM = (P, T, I, O, H, Π, M_0, D, W) be a GSPN defining the output-flow model. T = $\{t_e\}$, where t_e is an immediate transition (T_{im}). $p_{out,i}, p_e \in P$, where $p_{out,i} \neq p_e$. p_e ,is named final place. A token in place $p_e \in P$ models the conclusion of all service orders. $O(t_f, p_e) = 1$. $I(p_{out,i}, t_e) = n_i$, $\forall\ p_{out,i} \in P$, where n_i is the number of service orders related to the product i. $\Pi(t_e)=W(t_e)=1$..*

Next, the closure definition is proposed. The closure, represented by a transition connecting the *final* and the *start* places, provides a high-level representation of the production management system.

Closure Model – *Let CM = (P, T, I, O, H, Π, M_0, D, W) be a GSPN defining the closure model. T = $\{t_0\}$, where t_0 is an exponentially timed transition (T_{exp}). P = $\{p_{cin}, p_{cout}\}$. p_{cout}, p_{cin} are named final and start places, respectively. $I(p_{cout}, t_c) = 1$, $I(p, t_c) = 0$, $O(t_c, p_{cin}) = 1$ and $O(t_c, p)=0$, $\forall p \in P$, $p \neq p_{cin}, p_{cout}$. $D(t_c)=m$. $M_0=(1,0)$.*

As described in Section 2, a process is executed in phases. In the beginning of a phase, resources are allocated and, at the end, those are delivered. Besides, each phase has two buffers. The first one is named *source buffer* and the second is called *target buffer*. The model of a phase, denoted as Phase Model, is defined as follows.

Phase Model – *Let PM = (P, T, I, O, H, Π, M_0, D, W) be a GSPN model of a phase where, T = $T_{exp} \cup T_{im}$. $T_{exp} = \{t_r, t_f, t_p\}$ and $T_{im} = \{t_i\}$. The firing of transitions t_i and t_p models the beginning and the phase processing respectively. t_r models the time-to-repair and t_f models the time of the machine fail. P = $\{p_i, p_p, p_s, p_d, p_b\}$. A token in places p_i , p_p , p_s , p_d , p_b $\in P$ models a service order, a phase being executed, resources used for executing a phase, machine is down and the conclusion of a phase, respectively. $\Pi(t_i)=a$ and $W(t_i)=b$. $D(t_f)=\gamma$, $D(t_r)=\eta$ and $D(t_p)=\varphi$. $I(p_i , t_i) = I(p_p , t_p) = I(p_s , t_i)= I(p_p , t_f)= I(p_d , t_r) = O(t_i , p_p) = O(t_r , p_p) = O(t_p , p_s) = O(t_f , p_d) = O(t_p , p_b)=1$. $O(t , p_i)=0$; $\forall t \in T, t \neq t_r, t_f, t_p, t_i$. $I(p_b , t)=0$; $\forall p \in P, p \neq p_i, p_p, p_s, p_d , p_b$.*

With respect to system S again, let F and F' be two phases of the process *u*. The approach for getting the process model is the following:

1. Based on definition of phase model, draw each phase of process *u*.
2. For the phases F and F' of the process *u*, merge places in the following way:
   a. Merge places p_i and $p_{i'}$ if these places are related to the same source buffer of the phases F and F',
   b. Merge places p_b and $p_{b'}$ if these places are related to the same target buffer of the phases F and F',
   c. Merge places p_b and $p_{i'}$ if the phase F' is immediately executed after phase F,

d. Merge places p_s and $p_{s'}$ if equipment s = equipment s',

e. Execute step 2 for all phase models of the process u.

3. Merge places p_i related to the input buffers of the system S and denote the resultant place by $q_{in,u}$,

4. Merge places p_b related to the output buffers of the system S and denote the resultant place by $q_{out,u}$,

5. Execute steps 1 to 4 for u = 1, 2, …, n, where n is the number of products.

The Figure 3 shows places and transitions defined in the Phase Model.

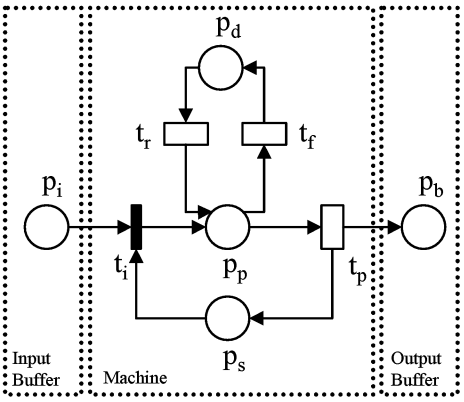


Figure 3: Phase Model

Based on the described approach just presented, the Process Models may be defined as follows.

Process Model - *Let PM' = (P', T', I', O', H', Π', M'$_0$, D', W') be a phase model. A Process Model PRM = (P, T, I, O, H, Π, M$_0$, D, W) is composed of phase models PM', such that PM'$_x$, PM'$_y$ ∈ PRM, T = ∪$^{\alpha}_{0}$ T', P = ∪$^{\alpha}_{0}$ P', P'$_x$, P'$_y$ ∈ P, where α is the number of phases, p_b ∈ P'$_x$, p_i ∈ P'$_y$, and p_b = p_i for any PM'$_x$ and PM'$_y$ sequentially combined; p_s ∈ P'$_x$, $p_{s'}$ ∈ P'$_y$ and p_s = $p_{s'}$ for any PM'$_x$ and PM'$_y$ share the same equipment s; p_b ∈ P'$_x$, $p_{b'}$ ∈ P'$_y$ for any PM'$_x$ and PM'$_y$ sharing the same target buffer; p_i ∈ P'$_x$, $p_{i'}$ ∈ P'$_y$ and p_i = $p_{i'}$ for any PM'$_x$ and PM'$_y$ sharing the same source buffer.*

Once obtained all the process models, the next step is to obtain the model of the whole system S. The approach to getting this model is the following:

1. Merge places $p_{in,u}$ and $q_{in,u}$ of the input flow and process u models, respectively. Denote the obtained place by $g_{in,u}$ the resultant place.

2. Merge places $q_{out,u}$ and $p_{out,u}$ of the process u and output flow models, respectively. Denote the obtained place by $g_{out,u}$ the resultant place.

3. Execute steps 1 and 2 for u = 1, 2, …, n,

4. Merge places p_s and $p_{s'}$ of process u and process u' models if equipment s = equipment s', for all places p_s and $p_{s'}$.

Figure 4 depicts a model of a system S. The makespam time (τ) is the time required for a system S to execute a number of service orders.

Let τ_{opt} be the minimal makespam time required

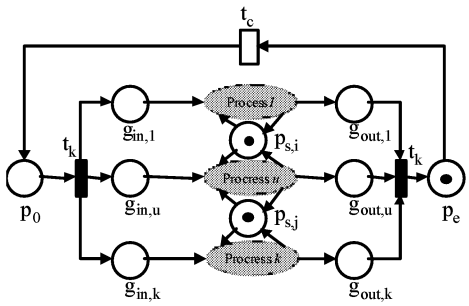


Figure 4: System Model

to execute N service orders using the resource constraint of the system S.

6. Resources Estimation

This section presents the method proposed for estimating the necessary number of resources for manufacturing items of a production line taking into account resources constraints.

In these models, the concurrent firing of transitions is constrained by the number of available resources. The goal of the proposed approach is to estimate the minimal resources of each available machine taking into account a given resource constraint. Therefore, the designer provides the maximal number of resources of each available machine.

To generate the results presented below, a complete Petri net model of the system depicted in Figure 4 was created. In Figure 5 a Petri net model of one cell is presented. Results were obtained through the use of TimeNET tool (Zimmermann 2001).

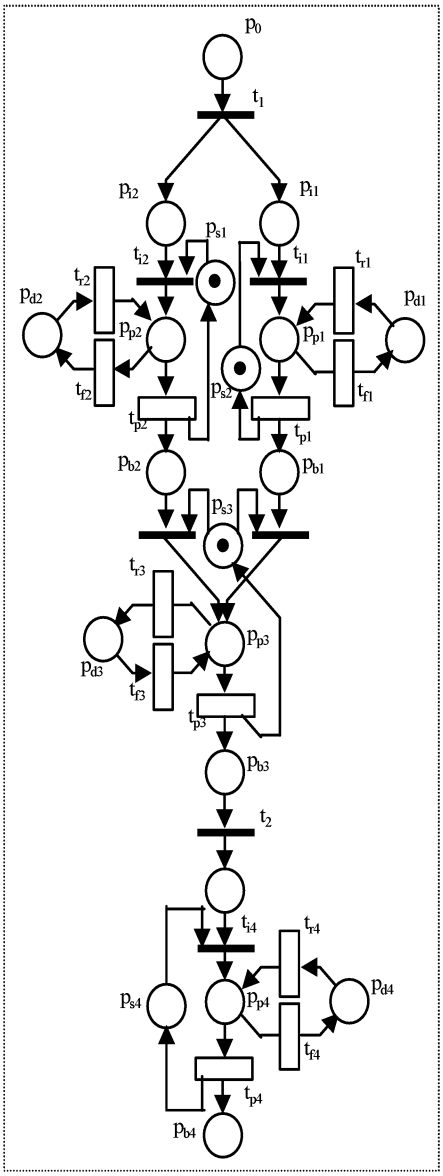


Figure 5: Petri Net Model for one cell

Place p_{s3} represents the availability of robot R1 to the input machines M1 and M2. The number of tokens in p_{s3} represents the capacity of R1 to serve multiple machines concurrently.

Fixing all net parameters and changing the capacity of robot R1 (marks in place p_{s3}), it is possible to analyze the effect of the availability of robot R1 to machines M1 and M2 in the makespam time of the system and to find a optimal number (or at least close to the optimum) that decrease the makespam. This is shown in Figure 6.

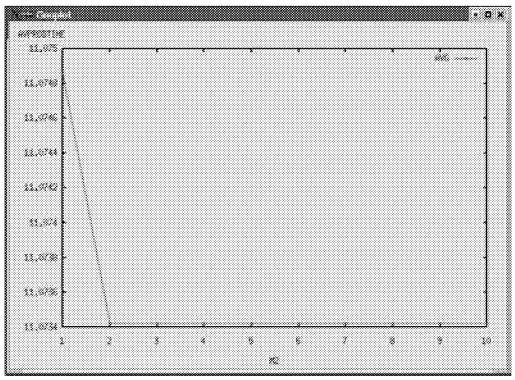


Figure 6: Time vs. Capacity

One should observe that the minimal makespam is obtained with only two resources available in machine M1.

The next step is to analyze the effect of the reliability of the machines in the makespam time. Through this analysis, it is possible to evaluate the increase of investment that may be considered for improving machine's reliability. For example, in net S, changing the time-to-fail of the machine M1 (transition t_{f1} of the net *S)* and analyzing the obtained result, one may conclude that after a given point (ex: 4000), the increment of the reliability of the M1 machine reliability does not result in a significant decrement of the makespam time. This is shown in Figure 7.

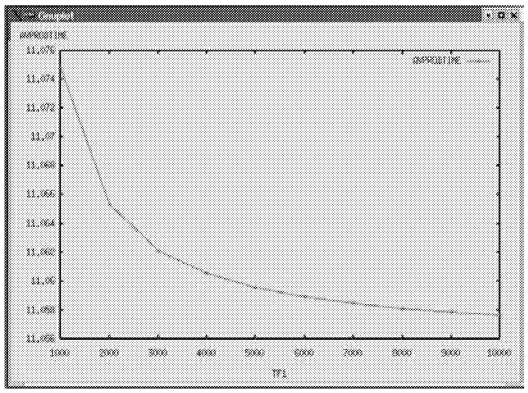


Figure 7: Time-to-fail

In the same way, it is possible to analyze the effect of the time-to-repair in the makespam time. Therefore, as expected, increasing the time-to-repair of machine M1 results in a increment of the makespam. This is shown in Figure 8.

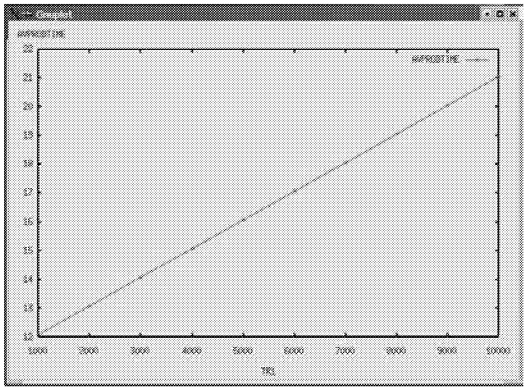


Figure 8: Time to repair

7. Conclusion

The work presented a GSPN based approach for modeling and estimation of resources for discrete production systems. The proposed model considers the production planning, plant's structure and machine's break down. However, one important issue taken into account was the estimation of the minimal number of resources and its cost required to execute the planning in a minimal makespam time. Dealing with this problem means to obtain the cheapest production cost in a minimal makespam time. In order to approach this problem, the model proposed has as input the system model and a set of initial markings that represents the cost constraints. However, the number of elements of this set (set of markings) may increase exponentially with the number of resource of the system. Hence, an approximation algorithm should be further considered.

References

Ayres, R.U. and Butcher, D.C. 1993. "*The Flexible Factory revisited*". American Scientist, vol.81, no.5, pp.448-459.

Murata, T. 1989. *Petri Nets: Properties, Analysis and Applications*. Proceeding of The IEEE.

Desrochers A. A.; Al-Jaar R. Y. 1995. *Applications of Petri Nets in Manufacturing Systems*. IEEE Press.

Marsan M. A.; Balbo G.; Chiola G.; Conte G.; Donatelli S.; Franceschinis G. 1991. *An Introduction to Generalized Stochastic Petri Nets*. Microelectronics and Reliability, 31, pp. 699-725.

Marsan M. A.; Conte G.; Balbo G. May 1984. "*A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems*", ACM Trans. Computing Systems, vol. 2, pp. 93-122.

Lee, D. Y. and DiCesare, F. 1987. *Integrated Scheduling of Flexible Manufacturing Systems Emplying Autometed Guided Vehicles*. IEEE Transactions on Industrial Electronics, vol. 41, N 6. December.

Elsayed E. A. and Boucher T. O. 1985. *Analysis and Control of Production Systems*. Englewood Cliffs, NJ: Prentice-Hall.

Zimmermann, A. June 2001. *A Software Tool for the Performability Evaluation with Stochastic Petri Nets*. Performance Evaluation Group, TU Berlin.

LATE PAPERS

EMERGENCY SERVICE: A GENERALISED FLEXIBLE SIMULATION MODEL

Paola Facchin
University of Padova
Via Giustiniani 3
35128 Padova PD, Italy
E-mail: epi@pediatria.unipd.it

Giorgio Romanin-Jacur
University of Padova
Stradella San Nicola 3
36100 Vicenza VI, Italy
E-mail: romjac@dei.unipd.it

KEYWORDS

Emergency service, Discrete simulation, Flexible model.

ABSTRACT

Any Emergency Service supplies first aid to outpatients; it may serve a hospital or a specific department. Its main characters concern high variability of patient arrivals, both depending on time and on randomness, and severe requirements for quick response, also in case of congestion.
The design and correct management of an emergency service concern structures, human resources, technological resources and working rules. System inner variability and multiplicity of driving decisions suggest to employ discrete simulation to analyse, check and dimension it in order to reach a satisfactory behaviour.
Building up the simulation model of a specific emergency service may be a heavy task: we need first the analysis of the structure and the functions, then the choice and estimate of useful parameters to describe the system.
In this paper we suggest and implement a general flexible simulation model, apt to correctly and fully describe almost all existing emergency services; it is based on common characteristics and conventions which are internationally accepted for pathology classification and on largely adopted assistance rules; it may be easily adapted to different situations for what concerns arrival patterns, structures, resources and working rules, by adjustment of few parameters. The model is implemented by specialised software Micro Saint, which presents a good compromise between abstraction level and user friendliness.

INTRODUCTION

The study of health services presents a dramatic development in the last years, especially connected to the aspect of cost analysis and containment (see for instance Boldy et al. 2002).
The aspect of hospital departments and services analysis, modelling and optimal dimensioning has received some attention, generally related to single cases.
In the present paper attention is devoted to the development of a generalised simulation model of emergency services, apt to be easily adapted to single situations.
The model is built up to obtain a detailed analysis of the behaviour of services, both already existing and at the planning stage, with the scope of determining the right dimension for what concerns utilised structures and resources.

CHARACTERISTICS OF EMERGENCY SERVICES

By definition, any hospital emergency service is devoted to supply first aid to outpatients. In general it may alternatively serve:

- a hospital or a group of hospitals in the same town, quarter or region,
- a department or a group of departments of the same hospital.

The service may supply aid to any outpatients, or to a specific patient class, according either to patients' pathologies (e.g., obstetrics, medical assistance, surgical assistance, orthopaedics, etc.) or to patients' specific characteristics (e.g., new-borns, children).
An emergency service of any type above is always characterised by high variability of patient arrivals, which obviously depends on intrinsic randomness of first aid request arising (e.g., because of accidents, of weather, etc.). Moreover arrival mean rate may depend on time (e.g., according to different intervals of the day, of the week, of the year). Although this inner variability, the effectiveness of the supplied service shall satisfy extremely hard constraints, as the response to aid requests shall be always quick and sometimes immediate, also in case of congestion, and every action performed shall be both effected with urgency and characterised by high accuracy.

THE PROBLEM

The general problem consists in supplying a high quality service at acceptable (when possible, minimum) cost. In order to obtain the above output any emergency service shall be correctly designed and managed for what concerns employed structures and resources. Structures include major treatment room(s), minor treatment room(s), beds, waiting room and possibly a short-term observation and assessment ward. Resources include technological resources (specific instruments) and human resources (medical doctors, nurses, and engineers). The service design determines the amount and type of structures and resources, the service management determines working rules, turns of duty, etc., of human servers.

DISCRETE SIMULATION AS A SOLUTION TOOL

As is well known, discrete simulation may constitute a useful tool to correctly dimension and manage production and service systems, because of its good level of detail and its absolute absence of bounds or constraints for what concerns analysing and checking the system behaviour, both in average and in exceptional working conditions. The negative side of simulation lays in the necessity of building a new model, or at least adapting an old one, often by

significantly modifying it, at every new application. The new built model shall then be suitably coded, implemented and checked, with consequent difficulties related to simulation languages, which may be expensive and are not always known and friendly to common users.
Discrete simulation may be usefully employed to correctly design and manage an emergency service if above difficulties can be successfully overcome.
In this paper we build up a generalised flexible model, able to correctly describe many different emergency services, both already working and to be implemented, by simply adjusting a limited amount of fundamental parameters of a common model structure.
We implement such a model by specialised software Micro Saint, which presents a good compromise between abstraction level and user friendliness, and therefore may be easily employed both by those who adapt the general model to the particular application and those who utilise the application for design or management.

THE MODEL

The suggested model describes in detail the patients' arrival, the "triage" operation, the patients routing and various treatments, which may include the use of external services (e.g., radiology, cardiology, etc.), the seizing and releasing of various structures, the human resource utilisation. A model is described with low detail in Fig. 1.
In every emergency service patients are examined by a nurse immediately after their arrival, and quickly classified on the basis of their "life parameters", according to their immediate life danger and the possible damage which they might suffer consequent to a delay in the assistance. The "triage" operations classifies every patient by means of a colour: "red" means severe life danger, so that assistance shall be immediate, "yellow" means a smaller life danger, so that a limited delay in the assistance (some minutes) does not worsen the patient's conditions; "green" means that a comparative long delay (e.g., thirty minutes) does not cause any damage; finally "white" means that no urgency is required; some countries (e.g., United Kingdom) use "blue" to classify patients in an intermediate position between green and white. Patients classification and related colour code determines the following routing, treatments and priority or possible pre-emption (interruption of a treatment being supplied to another less severe patient).
In the suggested model arriving patients are characterised by their conditions and related code. For every class (colour) the probability distribution of interarrival time and the variation of distribution parameters in time are set according to the statistics related to the particular application. Every considered class is doubled, as beside "normal" patients related to (random independent) accidents (exponential distribution with possible time dependent mean) we have also provided "special" patients related to exceptional events, like sudden epidemics, diseases connected with bad weather, etc., which may follow different distributions. Finally we considered a third type of "pseudo arrivals", describing urgent operations different from assistance to outpatients, to be performed at given times; such pseudo arrivals always include the exchange of orders, which is to be performed at every duty change, and possible checks to patient admitted in the short-term observation and assessment ward, when such ward is included in the emergency service.
At their arrival all patients are put in a queue and selected by the same unique filter (triage) which is ruled by one or more specialised nurses; "red" patients pre-empt all other patients, as their severe conditions are always evident.
Human resources include nurses, employed either in the triage or in the assistance, and one or more medical doctors; sometimes one or more engineers are employed to manage high level or specific instrumentation.
Doctors include a duty chief, who is responsible of the whole service, other doctors (possibly of different training level and experience), and "available" doctors, who may intervene within a short time (ten, twenty minutes), if called in case of severe patient congestion; finally other doctors may be employed in external services. Different positions, training levels and experience of single doctors may constrain the types of patients who may be treated and the types of operations which may be performed: as an example, for a red patients the chief must perform the first visit and a final check at the end of the treatments supplied, before patient release or admission to a hospital ward.
The duty period for doctors, nurses and engineers is generally either 12 or 24 hours; the number of people on duty is not necessarily the same for all duty intervals (e.g., more operators during the day then in the night). Other periods may be considered; in particular some operators may be on duty for shorter periods, in order to increase available human resources during "heavy" intervals of the day; finally people working in external services may follow different schedules.
Technological resources include specific instrumentation, like for instance ventilator and defibrillator.
As seen above, structures include major treatment room(s), minor treatment room(s), beds, waiting room, (possible) short-term observation and assessment ward.
Employment of structure and resources follows precise rules stated by the service managers; the rules determine constraints and priorities in allocating structures and resources to patients and state treatments to be supplied. The type and amount of structures, resources and operations requested (possibly by external services) and the times spent for the requested assistance operations are random variables depending on patient conditions: the related probability distributions are obtained from service observation, measurement and inference statistics.
The output of the model, useful for the service evaluation and for possible suggestion related to a better design or management, include all synthetic parameters reporting the behaviour of all queues (waiting time and queue length distribution), the utilisation of all structures and resources, the time spent in the system, both global and related to every patient class.

MODEL IMPLEMENTATION

The suggested generalised model was implemented by means of specialised software MicroSaint (Various Authors. 1998); MicroSaint is based on task network modelling,

where network are easily built up by a CAD method; an example may be seen in Figure 2, where the emergency service generalised model network is reported (some blocks related to tasks are represented by smaller symbols than usual because of space constraints). Main elements of networks are tasks, decisions and queues, whose parameters may be easily and clearly set, as may be seen in Figures 3. Moreover many built-in procedures rule simulation statistics.
The generalised emergency service model network, reported in Figure 2, includes 44 task blocks; from left we may note: a) a start block, necessary to start simulation; b) 19 generator blocks, which rule arrivals of patients of the four colours, doubled because of the two patient types (normal and special, as seen above), arrivals of external patients, and pseudo arrivals (exchange of duty and other); c) a triage block, which selects patients and decide their routing, which is bypassed by red patients; d) 3 blocks on the top concerning red patients' routing; e) 5 blocks concerning yellow patients' routing, which may include a path to external examinations; f) 2 blocks representing external examinations; g) 2 blocks representing possible change of patient class after first doctor's inspection or during therapy; h) 5 blocks concerning green patients' routing, which may include a path to external examinations; i) 1 block concerning white patients' therapy; finally l) 2 exit blocks representing admission to a ward or discharge.
In the generalised model implementation the network is almost fix, in that small changes are provided, mainly related to bypass some tasks, typically external arrivals to external examination, if the described operation is not performed in the particular service to simulate.
The "free" part, to be set for every application, concerns: a) all parameters ruling arrivals' probability distributions and time dependencies (coded inside arrival blocks' description tables); b) all parameters ruling decisions about resource utilisation, priorities end preemptions (coded inside inspection and therapy blocks' rule tables).
Once statistics and rules about the service to be simulated are known (generally by extraction from the service information system reports), parameter setting is very quick and easy also for non expert people; in fact instructions about setting operations could be collected in 15 pages.
Anyway, we noted that the list of parameters to be set constitutes itself a guide to emergency service modelling, and was much appreciated by medical users.

TEST ON AN ACTUAL APPLICATION

The generalised emergency service model was tested on an actual service, more precisely the Emergency Service of the Padova University Department of Paediatrics.
After fifteen days dedicated to an accurate analysis of the service register and a survey of operations, and a week dedicated to elaborate statistics, the model building required a very short time and the model implementation was ready at once.
The simulation revealed that the day turn was dramatically lacking in medical doctors: that was not evident to an external observer as usually help from wards satisfied necessities related to scarce human resources. Many useful suggestions were obtained about solving the problem without requiring excessive cost; modifications were effected on the doctors' duty schedule, by adding in the day turn one medical unit with full duty service (twelve hours) and one medical unit with (two plus four) service hours, conveniently distributed during the day in order to increase assistance capacity in correspondence with peak intervals (12-14 and 16-20).
The generalised model appears to be a useful managing instrument, which may be applied to different realities to improve the service performance, as it permits to detect critical points and consequently to suggest suitable correcting actions.

REFERENCES

Boldy D, Braithwaite J., Forbes I. 2002. "Evidence Based Management in Health Care: the Role of Decision Support Systems." ASHSA, Sidney.

Various Authors. 1998. "MICRO-SAINT 3.0: User's Manual." Micro Analysis and Design Simulation Inc., Boulder, Colorado.

BIOGRAPHY

PAOLA FACCHIN was born in Venice, Italy and went to the School of Medicine of the University of Padova, where she obtained the Laurea degree (equivalent to Master of Science) in Medicine and Surgery in 1977 with a thesis about detection of liver aldolase in hereditary fructose intolerance. She obtained qualifications from specialisation schools in Paediatric Clinics and in Hygiene and Preventive Medicine and she is Doctor in Development Sciences, epidemiological counselling. She was consultant in internal medicine, then fellow, then research assistant at the Department of Paediatrics of the University of Padova, where she is currently Associate Professor of Paediatrics and responsible of the Epidemiology and Community Medicine Unit, of the Regional Epidemiological Observatory of Paediatric Age Pathology, of Rare Sickness Register and of Ill Treated Children Crisis Unit. She currently teaches Community Medicine to the class of Medicine and Surgery, School of Medicine, University of Padova, and to many Specialisation Schools of the same University. Her current researches concern community medicine and health services.

GIORGIO ROMANIN-JACUR was born in Padova, Italy and went to the School of Engineering of the University of Padova, where he obtained the Laurea degree (equivalent to Master of Science) in Electrical Engineering in 1970 with a thesis about pulse modulation control system. He was fellow of the Italian Council of Researches in 1970-71; after military service in 1971-72 he was Research Associate of the same Council until 1980; in 1980-83 he was lecturer of Operations Research at the School of Engineering of the University of Padova; in 1983-2000 he was Associate Professor and since 2000 he is Full Professor of Operations Research at the same School. He currently teaches to the class of Management Engineering and to some Master and Doctorate classes in the Schools of Engineering, Medicine

and Agriculture. His current researches concern production scheduling in the fields of Industry and Services.

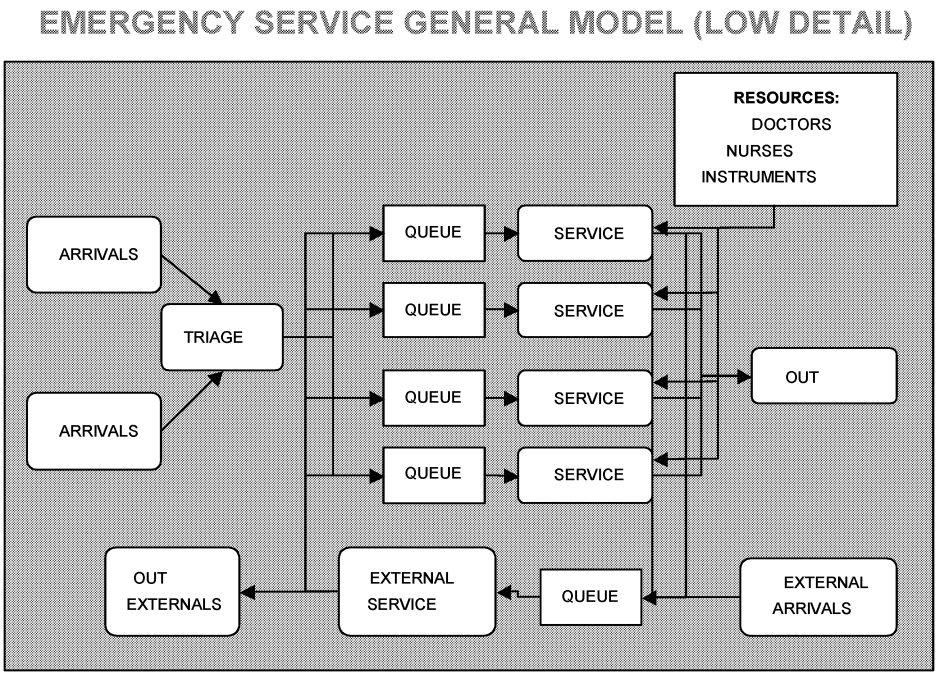


Figure 1

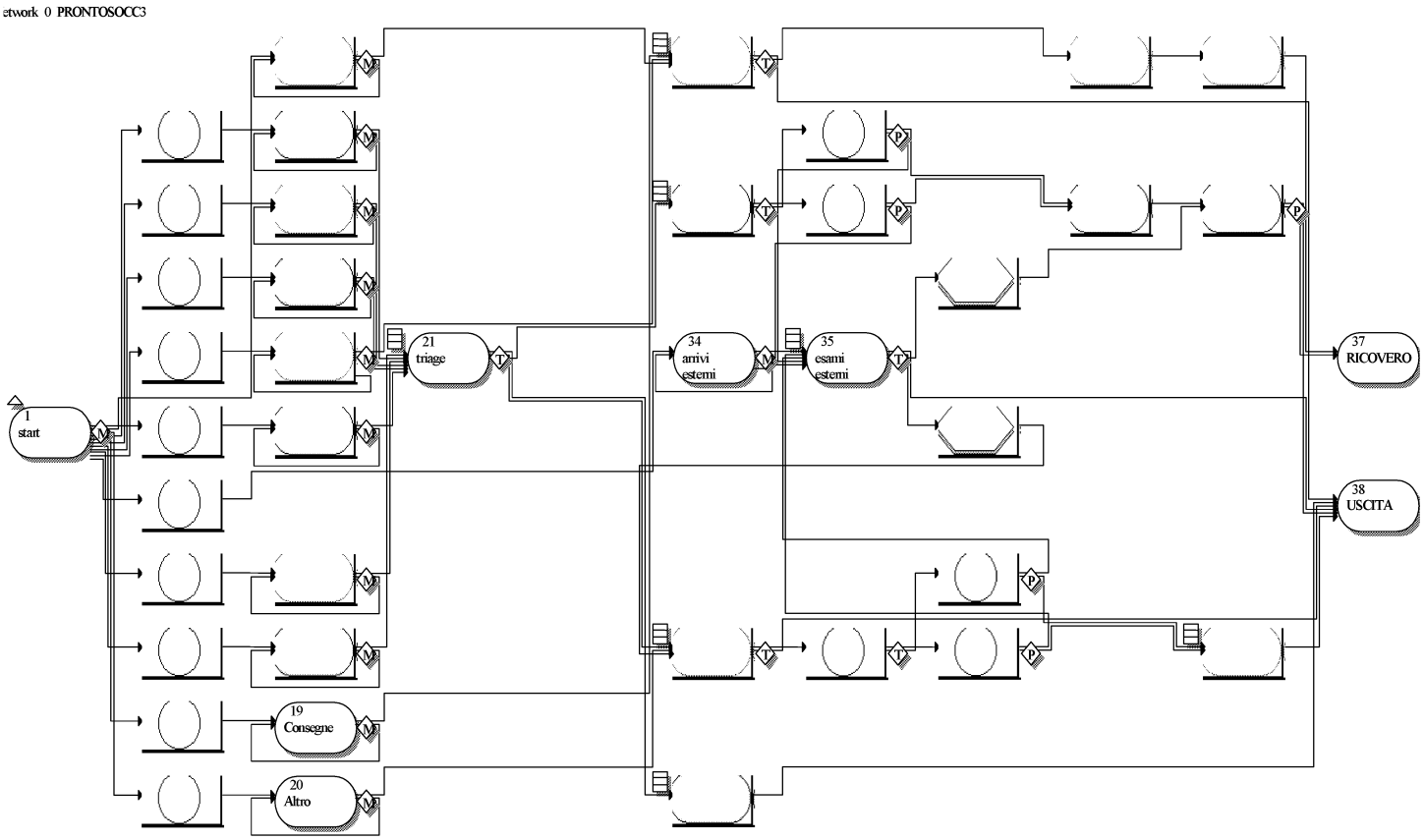
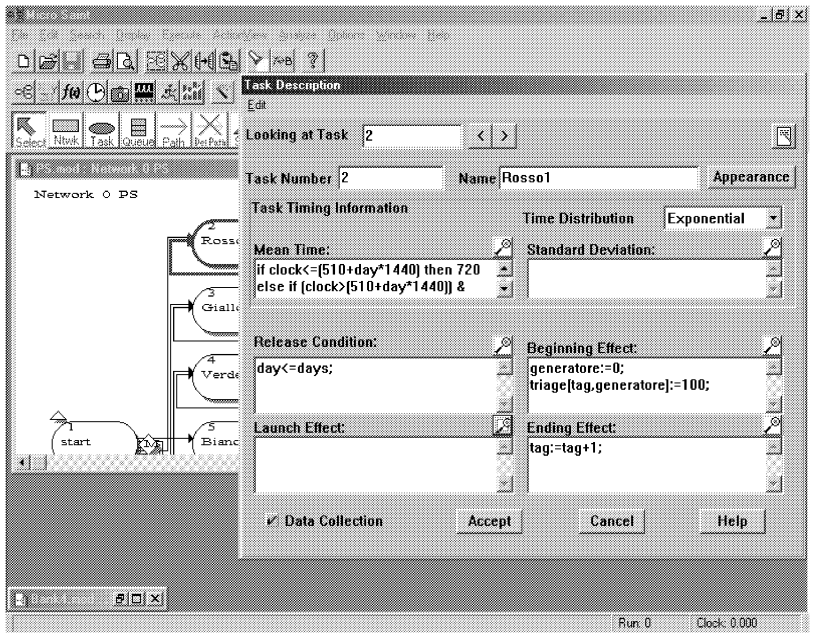
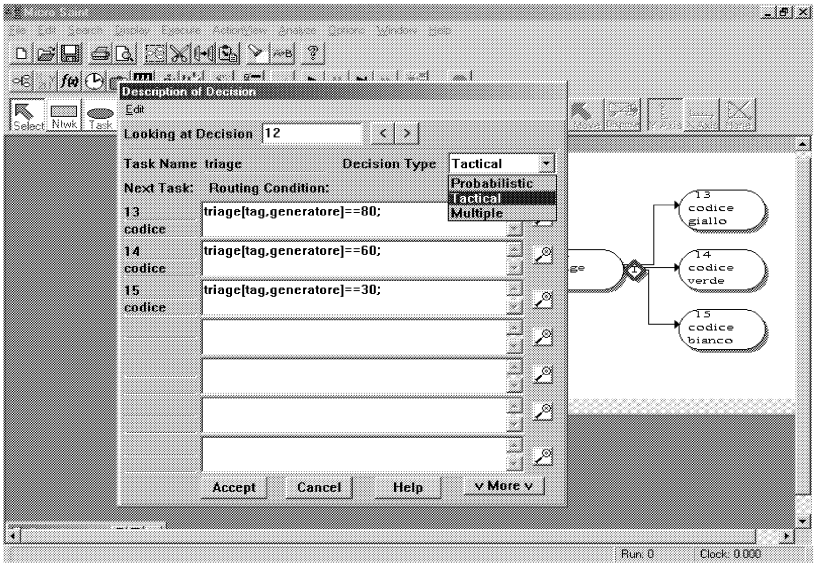
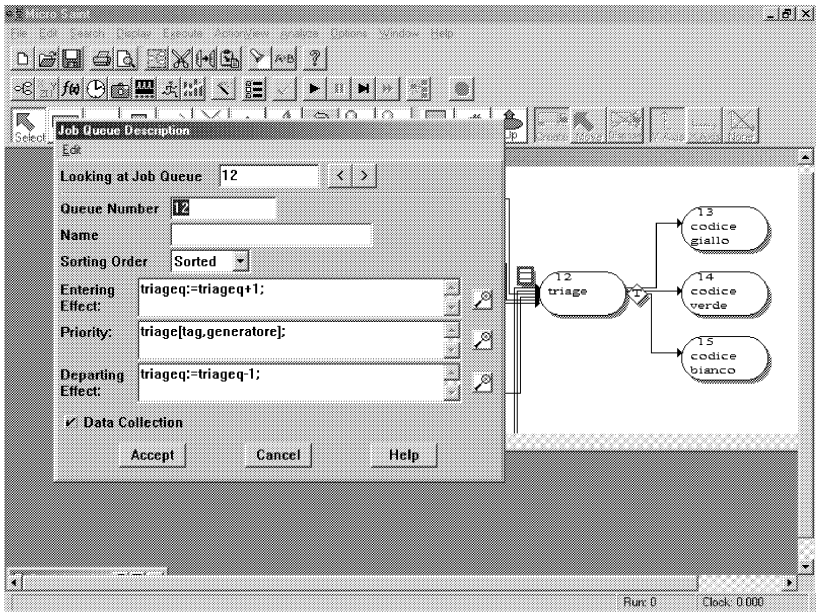


Figure 2: Micro Saint emergency service generalised model network







Figures 3: Example of: a) a task (red requests' generation), b) a decision (after triage), c) a queue (before triage)

Analysis of the "reliable criticalities" in a sintering plan: an application of Integrated Factors Method (I.F.M.)

Domenico Falcone, Gianpaolo Di Bona, Alessandro Silvestri
Department of Industrial Engineering
University of Cassino
Via G. Di Biasio n°43 - 03043 Cassino (FR) - Italy
e-mail: falcone@unicas.it; tel.: +39-776 299635; fax: +39-776 310812

Keywords: Affidabilità, RDB,FMEA, Allocazione.

Abstract
The techniques of the reliability allocation have been developed for the project development and for the operating and maintenance management of complex systems.
The maintenance management is an operating aspect consequent the project; on the contrary, the maintenance concerns project aspects.
Therefore, from the point of view of the operating maintenance, the allocation of the reliability must be considered as "allocation of the criticalities" or rather as a methodology which allows to identify the sub-systems responsible of the unreliability of the whole system.
The work intends to determine, through the application of the " Method of the Integrated Factors", the elements which are responsible for the reliable criticalities in a production sub-system, aiming
At the realization of sintered products. The allocation method introduced even if it supplied an analyses applicable to prototypal complex systems during planning, keeps general characteristics witch allowed its application also to the management and maintenance phases.
The production sub-system taken into consideration makes it possible to carry out the drying of dusts. The main element is a piston pump for the realization of vacuum, thus regenerating isopropyl alcohol.
The sub-system has been divided into functional blocks by means of the application of a Product Tree Analysis (P.T.A.) and a Reliability Block Diagram (R.B.D.).
Later, in order to quantify the allocation factors, a Failure Mode and Effect Analysis (F.M.E.A.) has been implemented.
We thought it convenient to fix the unreliability factor of the sub-system as target, taken from the scientific literature.

1. INTRODUCTION

The company success is a function of his customer satisfactions in terms of quality, price and service.
On this three aspect play a fundamental role two characteristics of the product: the reliability and the maintainability.
In engineering, safety factors and redundancies are often adopted to guarantee the component and installation reliability. Troubles or disasters suggest the right modifications in components or installations to guarantee the wanted system reliability.
In the last few years, some analysis methods are developing to estimate beforehand the components reliability, availability, maintainability and safety (R.A.M.S.), and to point out their weak points during the plan phase, in order to adopt the right actions.
The system reliability represents both the start and the finish point of R.A.M.S. Analysis:

- ✓ we are interested in the single sub-system reliability to obtain the whole system reliability (*reliability research techniques*);
- ✓ we can fix the system reliability value and then estimate components performances, obtaining the most critical sub-systems in terms of reliability (system design).

In the second case we speak of *reliability allocation techniques*; they permit to assign reliability parameters to the different system units, in this way the whole system reaches the established reliability target.
One of the principal advantages coming out from the adoption of reliability allocation techniques is the time and cost saving. In literature, there isn't a universal technique, correct for every system and every design state; so the research of the correct methodology for the observed system is a task of R.A.M.S. Analysis

2. THE PRODUCTION PROCESS AND THE SINTERING PRODUCTS

The dusts metallurgy is a particular technology, suitable for producing finished metal or metal ceramic pieces, starting from powders mixed through pressing and sintering operations. Generally, the process takes place in controlled atmosphere, and consists of an opportune keeping of the powders at such temperatures as to cause the conglomeration of all the mass, without reaching the fusion of the material.
The starting materials are metal carbides, usually tungsten carbide and cobalt powders.
The methods for the preparation of the metal powders can be divided into two groups:

- mechanical processes: processing at machine tools, crushing, grinding and gritting;
- chemical processes: thermic decomposition , condensation, reduction, precipitation and replacement.

Before being compacted, the powders are mixed in spherical mills, by adding right quantities of solid lubricant (paraffin) and a solvent of the lubricant (isopropyl alcohol). Alcohol must help the uniform distribution of the mass of the powders, forming a fluid mixture. After the removal of the solvent, the paraffin covers the particles of powder and helps their union and compacting on the following phase of pressing or extrusion.
The retrieval of the solvent used takes place through the drying of the mixture, carried out through the heating of the mixture with warm water.
The operation is effected under vacuum.
The following process of the pressing of the mixture can be of two types:

- mechanical: production of pieces in a finished shape and of medium-small size.
- hydraulic: production of box-shaped pieces and of medium-big size .

The extrusion process permits, as well as the hydraulic pressing, to obtain box-shaped and very long pieces. The products obtained are then delubricated and presintered , in order to give suitable capacities of mechanical resistance for the following moulding process.
The product, obtained through working by tool machine, is sent to the sintering phase.
The cycle is carried out in induction or resistance furnaces; and aims at changing the powder mixtures into a solid body:: (fig.1).

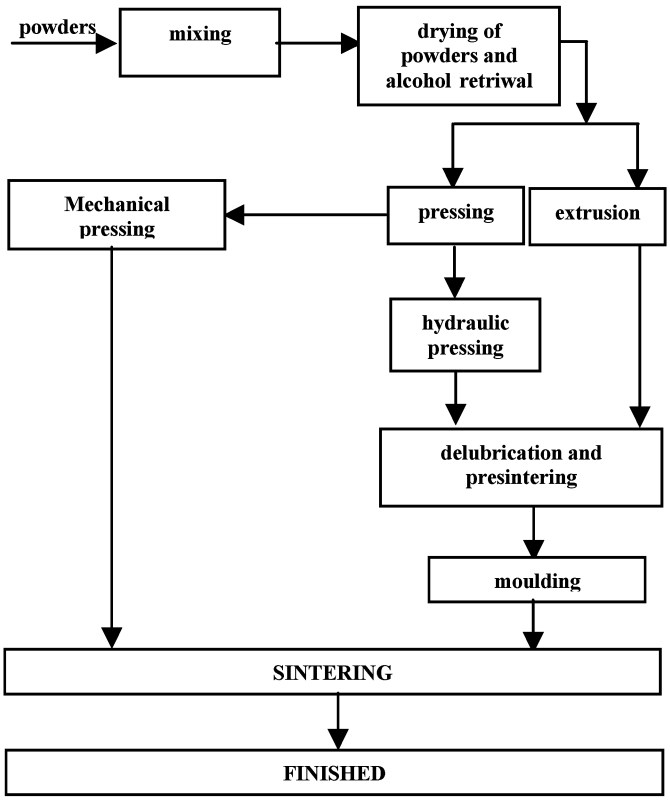


Fig.1 – F*unctional diagram of the productive cycle*

The fundamental parameters of such process are:

- time
- temperature
- shape and sizes of the particles
- surface state of the particles
- degree of compacting

For every type of powder there are some practical limits within which it is possible to make time and temperature changes.
Too low temperature values would cause too long and not economic times; on the contrary, high temperatures would help the rising of vaporisation phenomena, gas solubility as well as mixture fusion.
In general, when the temperature and the duration increase, even if they remain within the safety range, they aim at increasing the density, the density, the resistance at traction and extension of the compacted product.

3. FUNCTIONAL ANALYSIS OF THE PRODUCTION PROCESS.

The allocation methodology (M.F.I.) has been applied to the powder drying section, since it represents an important part within the production cycle. In fact, the lacking or partial retrieval of the lubricant would cause a difficult deposit of the paraffin on the powder particles, with consequent problems of compacting of the powders in the next phase of pressing.
The retrieval of the isopropyl alcohol is effected through a piston pump for the vacuum.
The lacking of physical redundancies causes probable breakdowns or malfunctions of the sub-system analysed to allow considerable inefficiency as to costs and quality of the process.
The piston pump for the vacuum, which needs the maximum reliability, shows the following mechanical characteristics

- motor power: 3 kW
- capacity: 100 m^3/h
- turns per minute: 250
- vacuum: 5,30 mbar
- total weight: 380 kg

The electric engine starts the rotation of the fly-wheel, which, through the camshaft, operates the connecting rod, the piston, the distributing rod and the slide valve.
During its run, while the piston inhales the gases of the cylinder from the slits, it discharges the gases previously inhaled during the reverse run through the valves.
The pump works in an intermittent way for about 20/22 hours a day.
During the remaining hours of inaction it is then possible to carry out operations of ordinary maintenance.
The lubrication of the sub-system is obtained through a pumping group formed by two pumps: a piston one and a gearing one.
The allocation of the reliable criticalities to the units of the sub-system required a preliminary study, whose main steps were:

- Product tree definition

- Reliability Block Diagram definition (R.B.D.)
- Preliminary Hazard Analysis
- Functional Analysis
- Functional – FMECA.

The RBD has allowed to divide the vacuum pump into six functional units, as follows (fig.2).

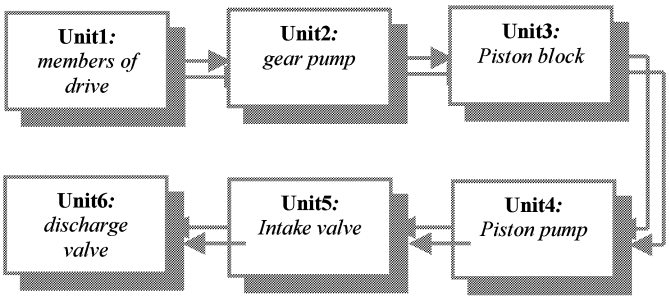


Fig.2 – *Reliability Block Diagramm*

The preliminary Hazard Analysis has allowed, besides, to determine two probable undesired events for which to fix some target hypotheses:

- loss of the functionality of the system (critical event)
- loss of the functionality of the system and of human lives (catastrofic event).

The functional blocks described have been the subject of a next functional analysis and FMEA.

The FMEA is a technique of support to the critical examination of the system, during all the phases of its life cycle.

The information supplied allows to determine the priorities for the process control and for the inspections , foreseen during the construction and installation.

These functional modules have allowed the estimate of the factors characterizing the M.F.I. method, in order to allocate the reliability criticalities of the system.

As regards the documents, it resulted advantageous to realize the FMEA using modules drawn up specially for the system under examination and prepared according to the purposes pursued. The information required by the module subdivided into columns, was:

a) Number of identification of the element of the system taken into consideration.
b) Denomination of the element.
c) Function carried out by the element
d) Ways and causes of failure
e) Way of working of the element
f) Effects of failure (local, superior, final)
g) Methods of pointing out of failure
h) Compensatory measures foreseen
i) Classes of gravity (critical and catastrophic)

4. INTEGRATED FACTORS METHOD (I.F.M.)

Some USA standards supply the guide lines for the development of a correct allocation model. They are:

- ✓ *generality;*
- ✓ *input data standardization;*
- ✓ *cheapness;*
- ✓ *definition of realistic and reachable requests.*

Starting from the above guide lines, a new methodology for the reliability allocation has been developed: *"Integrated factors method".*

The proposed reliability allocation technique has been thought-out for prototype complex systems during the pre-design phase.

Subsequently, the method has been changed, in order to permit its application in different design phases, when more and detailed information about components is known.

The new methodology has been initially developed for systems in series, prototype systems (the "series" hypothesis is in favour of security, in fact the mission failure happens when one unit breaks down).

Studying in detail the different methods, we have noticed the need to use opportune factors of influence.

These factors have to permit the discrimination among the system units.

Initially we have supposed the same technological level for the units and the same operative severity.

We have chosen the following factors and relative indexes:

- ✓ *Criticality index (C):* ratio between the number of sub-system functions that cause an undesirable event if not realised, and the number of total system functions;
- ✓ *Complexity index (K):* referring to the technological and constructive structure; possible values are: 0,10 for simple system; 0,20-0,90 for not very complex system; 1,00 for complex one;
- ✓ *Functionality index (F):* ratio between the number of total unit functions, and the number of total system functions;
- ✓ *Effectiveness index (O):* referring to the unit operative time; possible values are: 1,00 for whole mission time; 0,67 for continuous and long times; 0,33 for instantaneous times.

The evaluation of the above indexes, thanks to an Expert Judgement's help, permits to calculate the Global Index (*IG*), obtaining the system unreliability (and so the reliability) allocation:

$$(1)\ IG_i = \frac{K_i \cdot F_i \cdot O_i}{C_i} \quad \Rightarrow \quad IG\%_i = IG_i / (\Sigma_{i=1,\dots,n}\, IG_i)$$

$IG\%_i$: percentage global index relative to the sub-system$_i$;
IG_i: global index relative to the sub-system$_i$;
n: number of units.

Calculated the Percentage Global Index for each unit ($IG\%_i$), it is possible to allocate the system unreliability target ($U(t)$) to the unit$_i$ ($U_i(t)$):

$$U_i(t) = U(t) \cdot IG\%_i$$

Later, in a dynamical approach, we have passed from a quality definition for the *K* and *O* indexes to a quantity one:

- ✓ *Complexity index (K):* ratio between the number of parts of the unit and the number of parts of the whole system;
- ✓ *Effectiveness index (O):* ratio between the unit effectiveness time and the mission total time.

In order to apply the new method to more complex systems, characterized by components with different technology, we have introduced a *Technology index (S)* (S=0,5: traditional components; S=1: innovative components). To discriminate against electronic systems, against mechanical ones, characterized by the same complexity, we have introduced a further *Electronic Functionality index (E)* (E=1: completely electronic system; E=0,1: completely mechanical system).Finally, we have introduced an increase (M) for the *Effectiveness index (O)*, caused by a greater operative severity.

$$(2) \quad IG_i = \frac{K_i \cdot F_i \cdot O_i \cdot S_i}{C_i \cdot E_i}$$

5. I.F.M. APPLICATION TO THE SINTERING SYSTEM

The reliability allocation has been done by the application of the "Integrated Factors Method", first version (1). So we have stressed the versatile and modulate structure of IFM method.

The *S* (technology index) and *E* (electronic functionality index) parameters are not significantly for examined components. In fact, all the plunger pump units are completely electromechanical (*E=0,1*) and off-the-shell (*S=0,5* traditional components).

For each RDB units, we have estimated the allocation indexes by the application of a Functional Analysis-FMEA and thanks to an Expert Judgement's help.

Starting from the evaluation of the indexes:

- Complexity K;
- Critically C;
- Functionally F;
- Effectiveness O;

we have calculated:

- the Global index (*IG*) relative for the sub-systems for each top event (Critical and Catastrophic Event);
- the percentage global index (*IG%*) for each units;
- the unreliability allocated values U_i;
- the consequently reliability allocated values R_i,

The results obtained are exposed in tab.1

TAB.1: *Allocation Factors Values (Allocated Unreliability U_s=0.15)*

I.F.M. FACTORS	CRITICAL EVENT TARGET ALLOCATION					
	Unit 1	Unit 2	Unit 3	Unit 4	Unit 5	Unit 6
c_1: n° of critical functions	*18*	*12*	*6*	*5*	*6*	*5*
c_2: n° of unit functions	*30*	*14*	*10*	*15*	*18*	*8*
C = (c_1/c_2)	0,60	0,86	0,60	0,33	0,33	0,63
f_1: n° of unit functions	*30*	*14*	*10*	*15*	*18*	*8*
f: n° of plunger pump	*95*	*95*	*95*	*95*	*95*	*95*
F= (f_1/f)	0,32	0,15	0,11	0,16	0,19	0,08
O (E.J.)	1	0,67	1	0,67	0,67	0,67
K (E.J.)	0,9	0,8	0,5	0,8	0,4	0,4
IG	**0,47**	**0,09**	**0,09**	**0,25**	**0,15**	**0,04**
IG%	43,22%	8,41%	8,00%	23,17%	13,90%	3,29%
U_s target	**$3{,}00 \cdot 10^{-1}$** *(data banks)*					
U_i	12,97%	2,52%	2,40%	6,95%	4,17%	0,99%
R_i	87,03%	97,48%	97,60%	93,05%	95,83%	99,01%

The I.F.M allocation results, in terms of *IG%,* are the following ones:

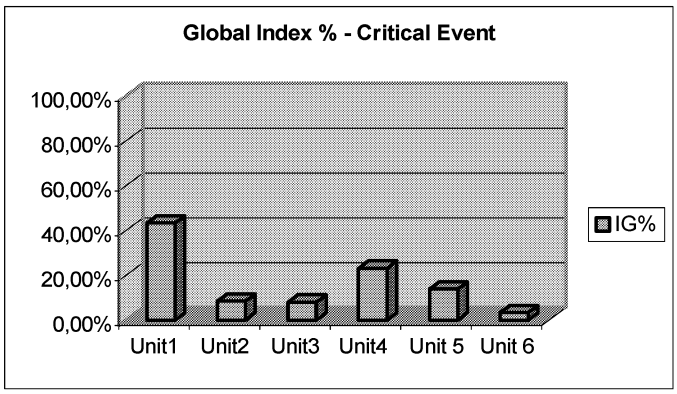


Fig.3 – *Percentage Global Index IG%*

Then, we have allocated the unreliability target $U_s=3{,}00 \cdot 10^{-1}$

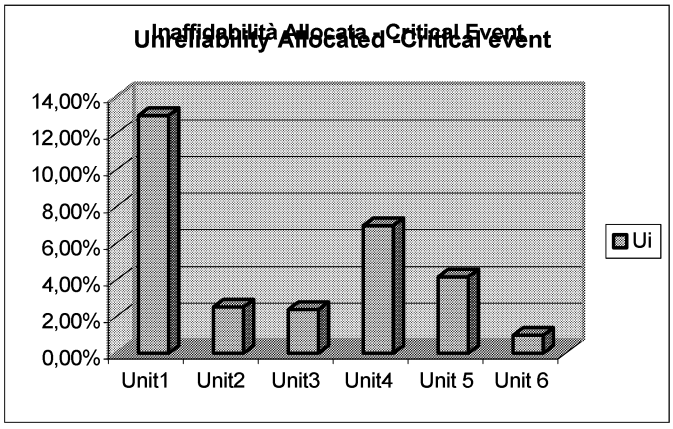


Fig.4 – Unreliability Allocated Values

Since the reliability values R_i are:

$$R_i = (1 - U_i)$$

it is possible to obtain the reliability allocated values for each units analysed:

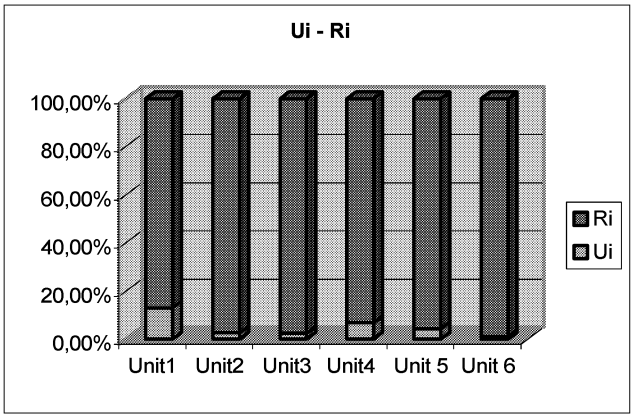


Fig.5 – *Comparison between R_i - U_i parameters*

Starting from an analysis of the results obtained, it is possible to notice:

- the proposed method is able to carry out a very particularized allocation, selecting among the different system units;
- the method is able to allocate the greater U_s value to the units that have an elevate Complexity K and Functionally F value indexes.

In conclusion the proposed method present the following advanteges:

- it is not necessary to know reliability data of similar units, so we can apply the method to innovative systems too;
- it reduced subjectivity through a complete indexes quali-quantity definition;
- it is possible to introduce or eliminate factors, adapting the method to the system and to the project phase;
- simple analytical treatment.

6. COMPARISON TO DATA BANKS

Subsequently, we have compared the IFM results with the reliability data, supplied to databanks or estimated by FTA (es. transmission system)

TAB. 2 – *Reliability databanks*

R_i DATABANKS						
	Unit 1 (FTA)	**Unit 2**	**Unit 3**	**Unit 4**	**Unit 5**	**Unit 6**
R_i	88,20%	95,58%	96,50%	94,80%	96,54%	94,25%

It is possible to notice that:

- the allocated reliability values are comparable to the supplied reliability ones;
- the units performance are respected (fig. 6).

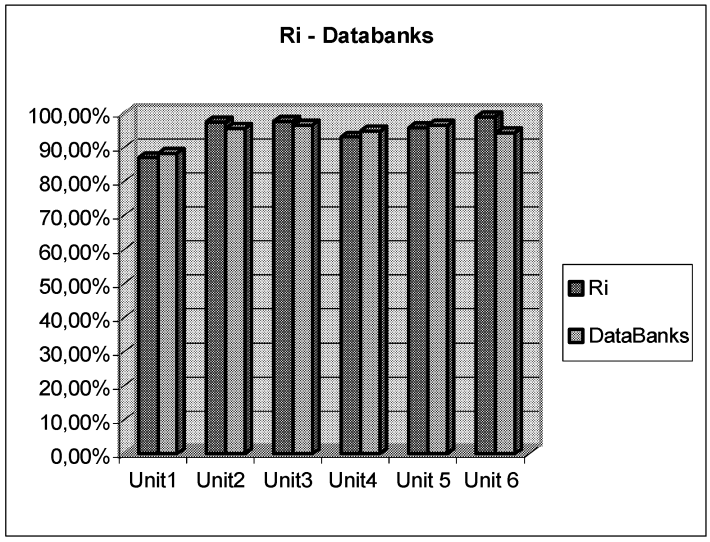


Fig.6 – *Comparison between R_i and databanks parameters*

These considerations permit to legitimate our method and its applicability to complex system, like plunger pump.

7. CONCLUSIONS

By validating the method through the application to a productive process, we have pointed out its advantages and its potentialities.

IFM allocates unreliability using a wide factors number, characterising each unit rightly. Introducing a product of factors, instead of a sum, we obtain a correct and discriminating allocation.

By the indexes evaluation, we can define a "reliability identikit" for each unit, and so assign the right reliability value. The identikit can be less or more detailed, by reason of the factors number and their definition level. The factors choice represents the method core, its dynamism and its modular structure, so it is not necessary to review the analytical treatment changing the system or its development phase.

The I.F.M. future developments will permit to design "Maintenance Plans" in those system and sub-system where the reliability databanks are poor or incompletely.

In these cases, the proposed technique will be able to point out the critical units where the more maintenance budgets are allocated.

8. REFERENCES

[1] Harold S. Balaban, Hal R. Jeffers: *The allocation of System Reliability - Vol. I: Development of Procedures for Reliability Allocation and Testing*, Technical Documentary Report, Armed Services Technical Information Agency Alington Hall Station, Virginia USA 1999.

[2] J.A.Boyd: *Allocation of Reliability Requirements: a new approach*, Proceedings Annual Reliability and Maintainability Symposium, IEEE (N.Y.), Las Vegas USA, 1992.

[3] D. Falcone F. De Felice V. Duraccio A. Russo V. Pesce: R.A.M.S. analysis for the cooling system of the conical nozze

of a plasma wind tunnel – International Conference Modelling and Simulation 2001 – IASTED 16-18 Maggio 2001 Pennsylvania USA.

[4] D. Falcone, A. Silvestri, G. Di Bona *Integrated Factors Method for Reliability Allocation: application to an Aerospace prototype project* Advanced Simulation Technologies Conference (Military, Governement and Aerospace Simulation 2003), Orlando, USA, 30 marzo – 3 aprile 2003

[5] D. Falcone, A. Silvestri, G. Di Bona Integrated Factors Method (I.F.M.): a new Reliability Allocation Technique" Software and Engineering Application IASTED, Cambridge USA, 4-6 novembre 2002

[6] V.D'Incognito: *Progettare il sistema manutenzione*, Ed. Franco Angeli.

[7] D.Kececioglu: *Reliability Allocation Apportionment*, Lecture Notes of Aerospace Engineering, University of Arizona, USA 1987.

[8] Military Standard 1629A: *Procedures for performing a failure mode, effect and critical analysis*, Department of defence 24 nov. 1980 USA.

[9] MIL-STD-721C: *Reliability*, Department of Defence USA.

[10] NASA Report: *Designing for Dormant Reliability*, Johnson Space Center (JSC) Guideline n° GD-ED-2207.

[11] NASA Report*: Improving Reliability and Writing Specifications*, Lewis Research Center (LRC).

[12] V.Nuzzolo: *Analisi di sicurezza per sistemi missilistici*, intervento al convegno "RAMS un approccio integrato", Associazione Centro Insulare per la Qualità, Roma 1996.

[13] F. Barbarino *Product Safety Engineering* - ISEDI 1990

[14] Kingery, Bowen, Uhlmann *Introduction to ceramics*, Wiley-Interscience, 1975.

[15] B. Bergam, B.Klefsjo, *Quality from customer needs to customer satisfaction,* Mc Graw Hill

[16] Thiokol Report: *Solid Rocket Motor Briefing*, Solid Propulsion Industry Action Group (S.P.I.A.G.) June, 1999.
George P. Sutton: *Rocket Propulsion Element*, John Wiley & Sons Inc.

BlueGene/L: A Powerful Platform for Simulation

Manish Gupta
IBM T. J. Watson Research Center

In this talk, we will give an overview of the BlueGene/L Supercomputer, being developed as part of a research partnership between IBM and the Lawrence Livermore National Laboratory. This massively parallel system of 65,536 nodes is based on a new architecture that exploits system-on-a-chip technology to deliver target peak processing power of 360 teraFLOPS (trillion floating-point operations per second). The machine is scheduled to be operational in the 2004-2005 time frame, at price/performance and power consumption/performance targets unobtainable with conventional architectures. We describe the challenges in designing a system software infrastructure that scales to 65,536 nodes, and how we have approached these challenges. We describe early experience with a 512 node BlueGene/L prototype running in our laboratory.

AUTHOR LISTING

AUTHOR LISTING

Ablan M. **367**
Acquaviva A. **30**
Ahern J. **206**
Aissani A. **448**
Akazawa Y. **316**
Alalouche T. **163**
Albertos P. **92**
Al-Din M.S.N. **216**
Alegre J.A. **19**
An X. **187**
Angrisani L. **323/429/434/465**
Arenas M.G. **378**
Arickx F. **24/180/262**
Aumeerally M. **458**

Balsamo S. **61**
Baron C. **99**
Bellamine-BenSaoud N. **378**
Benini L. **30**
Benitez I.J.. ..... **92**
Bertelle C. **355/360**
Bobeanu C. **513**
Bogliolo A. **30**
Bogunovic N. **221**
Boulos F. **442**
Bourcerie M. **489**
Briguglio S. **247**
Broeckhove J **24/180/262**

Cacciaguerra S. **169**
Candela C. **92**
Canonico R. **174/177**
Carvalho Brito A.E.S. **132**
Castañón-Puga M. **348**
Catret J.V. **155**
Choudhary A. **5**
Cioffi M. **257**
Coiro D.P. **291**
Colandrea M. **304**
Combettes S. **503**
Correcher C. **155**
Cristobal-Salas A. **348**

D'Antonio S. **465**
D'Apuzzo M. **429**
D'Arco M. **429**
Davidsson P. **143**
Dávila J. **367**
de Arcangelis L. **14**
De Marco A. **291**
De Neve H. **24**
Degenring D. **333**
Dewolfs D. **262**
Dhaene T. **24/180/262**
Di Bona G. **546**
Di Martino B. **247**
Diez J.L. **92**
Dvorák J. **203**
Dvořáková J. **203/375**
Dvornik J. **364**

Emma D. **174/177**

Facchin P. **541**
Falanga F. **434**
Falcone D. **546**
Farida B. **213**
Feliz Texeiro J.M. **132**
Ferretti S. **169**
Fischer C. **390**
Fogaccia G. **247**
Formisano A. **257**
Fortino G. **77**
Franke H. **192**
Freisleben B. **333**

Gambuzza A. **37**
Ghini V. **169**
Giambiasi N. **252**
Giannino F. **304**
Grasser T. **390**
Guiochet J. **99**
Gupta M. **552**

Hamerlain M. **163**
Hancke F. **24/180/262**
Hanne T. **83**
Hartinger M. **129**
Hasslinger G. **475**
Hellinckx P. **262**
Herbergs S. **148**
Hertkorn G. **129**
Hošáková Z. **203/375**
Hussain T. **199**

Iacono M. **403**

AUTHOR LISTING

Jagnjic Z. **221**
Jáuregui C. **348**
Jelic G. **364**
Johansson C. **143**
Jovic F. **221**

Kacsuk P. **6**
Katsaros P. **409**
Kazmierczak E. **420**
Kindler E. **122**
Klaudel H. **508**
Kocaman I. **88**
Kohlmeyer J. **328**
Krajewicz D. **129**
Křivý I. **122**

Lacomme P. **122**
Lacort J.A. **92**
Lattanzi E. **30**
Lazos C. **409**
Lee Y.T. **56**
Lehmann H. **148**
Leoncini P. **291**
Lerebourg S. **355**
Li Z. **298**
Liu W.Y. **521**
Lungu A. **107**
Lungu M. **107**

Mabed M. **489**
Maciel P. **531**
Mader W. **328**
Mancini E. **270**
Martone R. **257**
Marzolla M. **61**
Masi A. **434**
Massheder J. **304**
Mazzocca N. **270/403**
Mazzoleni S. **304**
McLean C. **56**
Mellado M. **155**
Merelo J.J. **378**
Mian M.S. **470**
Mieth P. **129**
Mikša Z. **203/375**
Moghal M.R. **470**
Mohan S. **453**
Molero X. **19**
Moles R. **341**
Moreno N. **367**
Moscato F. **403**
Munitic A. **364**

Namaane A. **252**
Nawawi M.K.M. **44**
Neu H. **83**
Nicolazzo M. **304**
Niijima K. **316**
Nketsa A. **503**

O'Regan B. **341**
Oberschelp O. **37**
Ojala L. **494**
Okada Y. **316**
Olivier D. **355/360**
Ottjes J.A. **279/293**
Otto P. **237**
Ozekeci S. **88**

Pasquino N. **69**
Pathak C. **420**
Pavard B. **378**
Penttinen O.-M. **395/494**
Peter S. **148**
Pistol C. **107**
Prevost G. **360**
Puig D. **155**

Qeli E. **333**
Quintero R. **367**
Qutaishat M. **229**
Rak M. **270**
Rateb F. **378**
Renken H. **192**
Ringel J. **129**
Roccetti M. **169**
Rodriguez-Diaz A. **348**
Roessel C. **129**
Romanin-Jacur G. **541**
Russo W. **77**
Rust C. **521/526**

Saffiudin M. **453**
Sahili J. **387**
Santonja V. **19**
Sarstedt S. **328**

AUTHOR LISTING

Schamberger S. **526**
Scheideler P. **192**
Schiano Lo Moriello R. **323**
Schmidt A **192**
Schnitter S. **475**
Selberherr S. **390**
Shamey M.R. **199**
Shao G. **56**
Silva E.A. **206**
Silva M. **531**
Silva W. **531**
Silvestri A. **546**
Sitte R. **298/309/459**
Sivalenka A. **453**
Sklenar J. **117**
Sliskovic M. **364**
Smail R. **448**
Stappert F. **521/526**
Strelen J.-C. **51**
Stuer G. **262**

Tahar R.M. **44**
Tan B. **415**
Tanguy A. **122**
Tatur K. **309**
Thanh C.B. **508**
Torella R. **270**
Torregrosa I. **19**
Tranouez P. **355/360**

Urbanek J.. ..... **203**
Urbánková V. **203/375**
Uzcátegui M. **367**

Vadursi M. **465**
Van Landeghem H. **513**
Veeke H.P.M. **279/283**
Ventre G. **174/177/465**
Veselov A.A. **420**
Villano U. **270**
Vlad G. **247**

Wagner P. **129**
Wagner S. **390**
Wahl A. **333**
Wernstedt F. **143**
Wiechert W. **333**
Wileden J. **206**

Xiang Y. **187**

Zaitsev D.A.. ..... **485**