Design and Analysis of Computer Experiments with SimExplorer

Thierry Faure, Nicolas Dumoulin, Florent Chuffart, Guillaume Deffuant
CEMAGREF – LISC
24 avenue des Landais - BP 50085
63172 Aubire cedex
email: firstname.lastname@cemagref.fr

KEYWORDS
Design of Experiment, Sensibility Analysis, Simulation, Computing on Demand, Framework, IDE, API.

ABSTRACT

The attendees will acquire the concepts and some practise of SimExplorer, allowing them to use it autonomously for their own needs. SimExplorer is a framework that includes an API[1] and an IDE[2] dedicated to design and analysis of computer experiments. SimExplorer combines blind computing on demand service access, traceability and reporting features and sensibility analysis tools. This tutorial exemplifies a typical use of SimExplorer, performing a global sensitivity analysis of a given individual-based predator prey model.

CONTEXT

The constant growth of computing power gives the possibility to run simulations of an increasing complexity. This is particularly the case for environmental dynamics modelling, which often involves coupled heterogeneous sub-models (biological model coupled with meteorological or hydrological models for instance) or individual-based models which represent explicitly individuals of a given population (human, plants, animals, ...). One can expect that this growth of the model complexity will provide a higher accuracy and a better understanding of the phenomena at stake. Unfortunately, this expectation is very often deceived, because the computer models become themselves so complex that it is more and more difficult to understand them, and to evaluate the confidence to give to their results. Consequently, their use as decision supports is seriously weakened, because decision makers need understanding and confidence to ground their decisions. This situation suggests to consider these models exactly like complex natural phenomena, that scientists try to understand. The most important difference is that to design experiments on a computer model is generally easier than on a natural phenomenon [5].
Generally, it is necessary to develop a specific application to "explore" each model. This application should often manage the generation of well controlled initial conditions: specifically distributed lists of values, for example: spatial distributions, networks of interactions with specific properties (small world, scale free, ...). Such an application represents heavy investments, and much of the scientific quality relies upon it. The motivation behind SimExplorer [3] is to invest once for all in a tool that can be used to design and execute properly experiments on any model. The goals are multiple:

- to externalise the development of the model exploration, in order to make available some generic methods and tools which can be applied in most of the cases for any model to explore;
- to favour the reusability of available components, and therefore lower the investment for good quality model exploration applications;
- to facilitate a quality insurance approach to model exploration using traceability features.

SIMEXPLORER FRAMEWORK

SimExplorer is developed with a component based architecture in Java (OSGI[3][4]). The graphical user interface (GUI) is based on the NetBeans Rich Client Plaform, which includes some pluggable tools to help the development of experimental designs, and language editors (java, groovy, ruby). Figure 1 illustrates the basic components of SimExplorer architecture.

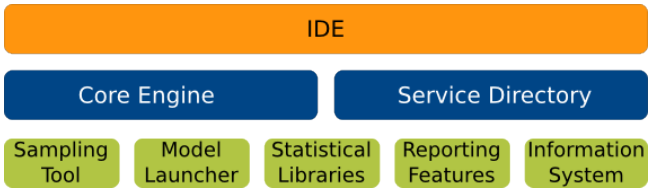

Figure 1: SimExplorer architecture.

Sampling Tool – An experimental design aims to explore a model upon a specific domain. The Sampling Tool is in charge of selecting elements of this domain according to the selected design.

[1] API – Application Programming Interface.
[2] IDE – Integrated Development Environment.
[3] OSGI – Oriented Service Gateway Interface.

Model Launcher – Model exploration requires expensive model execution in CPU and/or memory. The model launcher ensures the distribution of this task over cluster or grid architecture.
Statistical Libraries – Numerical exploration of model produces large amounts of data, that need to be aggregated or graphically displayed to be interpreted. Third party libraries could be plugged in the SimExplorer runtime to improve the available analysis tools. SimExplorer integrates the access to the R[4] software and Dakota[5], and other numerical tools integration is planned.
Reporting Features – In order to disseminate and communicate about and experiment, SimExplorer offers reporting features. Reports are elaborated by collecting data and metadata from the information system, and organising them into a document format (web, pdf . . .).
Information System – The information system stores the steps and results of the exploration, and can be synchronized with a server application that allow to share data with other users. This component is relevant to garantee the reproductibility of any exploration and to improve the collaboration and the publication of results.
Service Directory – SimExplorer is based on Service Oriented Architecture. Consequently, the Service Directory component ensures service discovering through the network.
Core Engine – The core engine drives services according to the design. It uses Service Directory to discover service and access to their methods. For example, figure 2 illustrates the dialogue between de Sampling Tool and the Core Engine. the Core Engine address the Sampling Tool using design specifications. In response, the Sampling Tool gets asked scenarios that will be used to evaluated the model.

Figure 2: Sampling Tool and the Core Engine illustration.

IDE – The integrated development environment (IDE) offers graphical user interface, and programming script language. Its purpose is to design an exploration application, that is achieved by building a workflow of treatment components. The default workflow proposed is composed of an exploration loop component and a global output processing component. The exploration loop needs to define some exploration factors with their definition domains, and the design of experiment that will be used to sample the factors. At runtime, this exploration loop component will loop for each samples values of the factors defined on a list of nested components to integrate the model to explore. First, input processing components are needed to map the sample values of the factors to the input data of the explored model, and to generated related input files. Then, informations needed for launching the model are filled in an appropriate component depending of the technical access to the model (binary, java archive, webservice, . . .). And finally, the last step proposed for this exploration loop is intended for processing the output of the model execution, that is for gathering results from output files and storing them in SimExplorer runtime for following steps.
A screenshot of the IDE is showed on the figure 3. The left side of the window allows to visualize and edit the workflow of the exploration application. In the middle, editors are available according to the component selected in the workflow. The right side of the window presents several tools to define the data used in the exploration runtime.

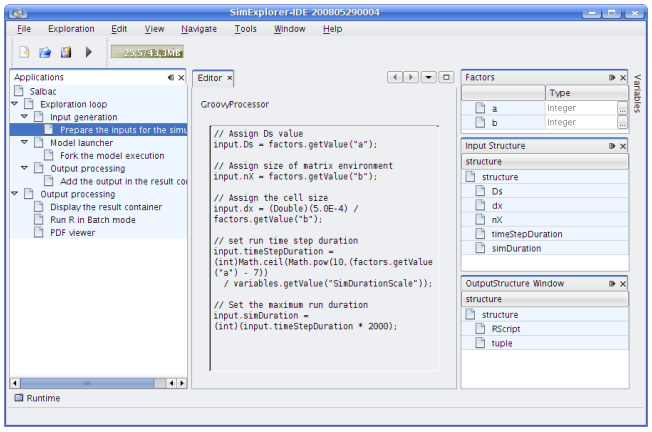

Figure 3: SimExplorer IDE.

SimExplorer platform is tested by the exploration of complex models from various applications fields (bacterial colony model, savanna and language dynamics) in the European project PATRES (NEST-043268) context.

OUTLINE OF THE TUTORIAL

The attendees will acquire the concepts and some practise of the software, allowing them to use it autonomously for their own needs. The tutorial begins with an overview the the SimExplorer software. Then, it introduces the main methods of design of experiment. Next, it explores the predator-prey model described in the next section to discover pratically the SimExplorer IDE features. After which, it gives a try to the client-

[4]R – Free software environment for statistical computing and graphics. http://www.r-project.org/
[5]Dakota – Design Analysis Kit for Optimization and Terascale Applications. http://www.cs.sandia.gov/DAKOTA/

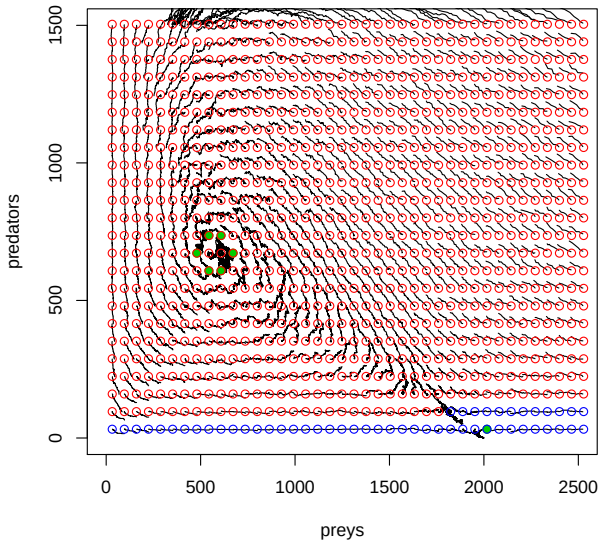

Figure 4: attractor (green point), trajectory (black lines) and partition concepts (red and blue circles) for a given state space.

server information system, to the distributed execution feature and to the report generation tool. Finally, audience builds an exploration upon his own brought model.

HAWK AND DOVE PREDATOR PREY MODEL

Hawk and Dove Predator Prey model (HDPP) is an individual-based version of a particular *Lokta-Volterra predator prey model* [1, 2]. It integrates individuals – predators and preys – that are spatially located on a 2D-grid. At each time step, all predators make several random moves, of a limited range. If they visit a site of a grid where a pry is located, they capture the prey. Then, they can share or struggle for the prey with the other predators located within a given neighbourhood. Their attitude (aggressive or cooperative) is defined for the time step. But after each time step, the predators can change their attitude by observing the success of the attitude of their predator neighbours during the last time step. They tend to adopt the most successful attitude.
The HDPP model analysis reveals that, depending on the values of its parameters, the model dynamics can lead to the extinction of the predators, or a single attractor, or two attractors, or one attractor and the extinction. Figure 4 exemplifies the attractor, trajectory, partition concepts for a given state space. State space is divided into a regular grid. Each cell is characterized by the trajectory starting from its centre. When the trajectory get out of the cell, the new cell where the trajectory is, is associated to the starting cell as its next. Thus, the state space is represented as graph where circuits are attractors, elements that compose two path to the same attractor are in the same partition.
Individual behaviour of the HDPP model is classically characterized by the mortality and birth rate of each species and the prey carrying capacity parameter. Due to the *hawk* and *dove* tactic introduction, model becomes sensible to the gain parameter. Finally, individual based model constraints to introduce a neighbourhood parameter.

CONCLUSION

The SimExplorer software will be freely available in September 2008 under GNU GPL licence on http://www.simexplorer.org.

REFERENCES

[1] Pierre Auger, Rafael Bravo de la Parra, Serge Morand, and Eva Sanchez. A predator-prey model with predators using hawk and dove tactics. *Mathematical Biosciences*, 177&178:185–200, 2002.

[2] Pierre Auger, Bob W. Koi, Rafael Bravo de la Parra, and Jean-Christophe Poggiale. Bifurcation analysis of a predator-prey model with predators using hawk and dove tactics. *Journal of Theoretical Biology*, 238:597–607, 2006.

[3] Thierry Faure and Guillaume Deffuant. SimExplorer: A software tool for programming and executing experiemental designs on complex models. In *SCS International Conference on Modeling and Simulation - Methodology, Tools, Software Applications (MS-MTSA 06)*, pages 218–225, 2006.

[4] Jan S. Rellermeyer, Gustavo Alonso, and Timothy Roscoe. R-osgi: Distributed applications through software modularization. In Renato Cerqueira and Roy H. Campbell, editors, *Middleware*, volume 4834 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2007.

[5] J. Sacks, W. Welch, T.J. Mitchell, and H.P. Wynn. Design and analysis of computer experiments. *Stat. Sci.*, 4:409–435, 1989.