SUBMODELS, MODELS, EXPERIMENTS AND PROJECTS

Antoni Guasch

Departament ESAII
Universitat Politècnica de Catalunya
Diagonal 647 - 2 planta
08028 Barcelona (SPAIN)
guasch@esaii.upc.es

Depart. of Computer Science
California State University
Chico, CA 95929-0410

Xavier Montfort
Departament ESAII
Universitat Politècnica de Catalunya
Diagonal 647 - 2 planta
08028 Barcelona (SPAIN)
montfort@esaii.upc.es

ABSTRACT

SIMBIOS (SIMulation Based on Icons and ObjectS) is an object oriented continuous simulation environment written in Smalltalk/V Windows. Models are defined graphically on the screen using a block diagram format. In this paper, the project concept is introduced. It is the element responsible of the simulation project management. A project is composed of a set of submodels, models and experiment. The submodels are grouped into models and every model has one or more experiments associated to it. A submodel can be simultaneously present in more than one model and any change made to the parameters of that submodel will be automatically seen in all the models where the submodel is present. This capability increases the flexibility of the simulation environment.

INTRODUCTION

Object-oriented programming (OOP) became available with the release of SIMULA in 1967 (Birtwistle 1973). However, it took about twenty years to appreciate the virtues of the paradigm. SIMULA and more recent object-oriented languages have been extensively used in the area of discrete simulation. Although this methodology had little impact in the area of continuous simulation in the past, the use of OOP techniques in continuous simulation is not new. It should be mentioned the work of Sim (1975) and Helsgaun (1980) who developed a SIMULA class library extension each which expand the functionality of SIMULA to allow continuous as well as combined simulation. See Kreutzer (1986) for a good review of object-oriented simulation topics.

SIMBIOS can be classified at present as an object-oriented continuous *simulation software* prototype. This clearly differentiates it from a *modeling software* (Cellier 1991a). While this research work has been centered in the area of object-oriented simulation, other groups, and specially the simulation teams at the University of Arizona and Lund Institute of Technology, have focus their own in the area of object-oriented modeling. See Cellier (1991b, 1992), Andersson (1989) and Mattsson (1991) for further references.

Up to now, the main emphasis on the SIMBIOS related research has ben in the area of the simulation engine architecture of the description of the model to be exercised and of the simulation engine itself. The main motivation that has propelled this work has been the need of a more flexible, reliable and in-house simulation environment that could support further work in the areas of system modeling and knowledge-based simulation.

SIMBIOS supports without restrictions hierarchical run-time simulation models. However, it has to be noticed that hierarchical modeling does not necessarily imply hierarchical simulation since the hierarchical model can be expanded for flat execution (flat simulation model). In fact, while it is generally assumed that hierarchical object-oriented modeling interfaces are desirable there is no agreement about the need of object-oriented run-time models and object-oriented engines (object-oriented simulation). The detractors advocate that the inefficiency of the run-time object message passing is critical in continuous simulation due to the tight coupling between submodels. This is certainly true in the Smalltalk SIMBIOS simulation environment prototype. However, languages with object-oriented features such as C++ give higher priority to run-time efficiency criteria that prevail over methodological purity that could slow down the execution. Therefore, C++'s object-oriented constructs can be used without major concern for their run-time efficiency (Beck 1990).

USER INTERFACE

The user interface of SIMBIOS is based on the intensive use of windows and icons. The simulation environment has specialized windows for the different tasks. For example: an icon-based window for the model definition, a window for the experiment specification and a window for graphical and numerical results (Guasch 1991).

The SIMBIOS icon-oriented interface is similar to graphical interfaces available in software systems such as EASY5 and MATRIX$_X$ (Simulab). Each element of a model, let say, a primitive element such as an integrator, or a submodel such a distillation column, has its own icon. A model is constructed by placing the icons on the screen and connecting them as required. The graphical modeling interface is based on the concept of block diagram modeling (Cellier 1991a).

The graphical modeling window behaves like a virtual screen, i.e., the virtual screen is as large as needed and the physical screen can move arbitrarily over the virtual screen. Also, the user will be able to collapse a set of interconnected icons into a single icon. This feature introduces hierarchy at the graphical (modeling) level but not in the simulation model. SIMBIOS supports hierarchical simulation, however, the mechanisms to create a submodel class from a set of graphical interconnected icons have not been implemented yet because Smalltalk/V Windows does not allow the dynamic creation of classes in run-time mode. The SIMBIOS hierarchical simulation capabilities have been tested with hand made hierarchical models.

A SIMBIOS graphical model is not the static representation of a model that requires further analysis, preprocessing and compilation before execution. In SIMBIOS, a submodel or primitive element instance is automatically created when its associated submodel icon is picked and placed in the screen. Therefore, the model can be directly executed since the traditional sequence of editing, preprocessing, compiling, linking and simulating is avoided. Moreover, model analysis is on-line with the graphical editing and it is performed each time that an icon is inserted or connected. For example, implicit loops are detected and graphically emphasized on line with the model set-up.

The example in figure 1. shows the format of the icon-based graphical modeling interface. It models a hydroelectric turbine automatic control system.

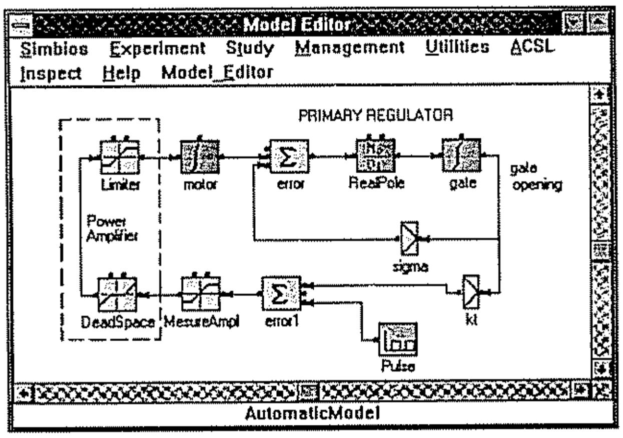

Figure 1. Hydroelectric turbine automatic control system model (*AutomaticModel*).

Figure 2 shows the system response gate opening versus time.

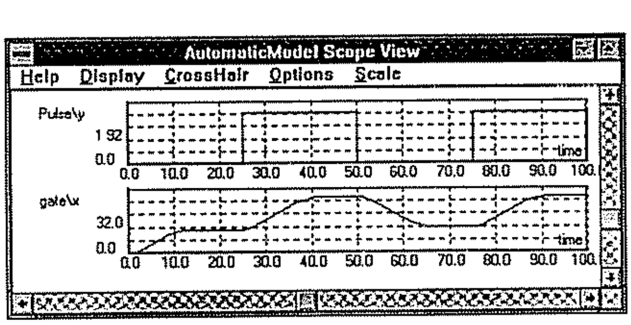

Figure 2. Gate opening versus time.

The current SIMBIOS prototype runs in the Windows 3.0 environment. Therefore, the simulation model or the graphical and numerical simulation results (in general, any SIMBIOS window) can be directly imported into word processors such as Microsoft Word for Windows (WinWord) that work in the Windows environment. This easiest the task of generating project documentation.

The SIMBIOS model and related experiment can be directly translated into ready-to-run ACSL code. Thus, SIMBIOS can be used as a block-oriented model editing interface for ACSL and faster execution speed can be achieved.

MODULARITY AND HIERARCHICAL APPROACH

In medium to large simulation projects, simulation users can be interested in studying or working with more than one model at a time:

1) Sometimes, the simulation user needs to work simultaneously with different models, i.e., models that represent the system behaviour at different levels of complexity or models that emphasize different aspects of the system. This usually happens when the user has to find a model that best suits his particular needs.
2) In large-scale modeling and simulation it is usually convenient to represent and simulate the model in a hierarchical manner. The model coding is naturally performed bottom-up and it is always convenient to execute experiments for every level of the hierarchy. Therefore, every submodel can be seen as a model if a particular experiment is applied over that submodel.
3) For validation purposes, it is very useful to compare real data collected from the physical system with data available from simulation. Therefore, beside the simulation model, we would like to handle the real data in another active model.

SIMBIOS can hold several active models at a time within the system. In this context active model means: ready-to-run model although it can not be executed concurently with other models. For example, let us suppose that the goal of a project is to design a controller for a non linear process. In this case, a first simulation model of interest can be the non linear process, and a possible experiment is the identification of the process parameters. The second model can be the whole system and the objective of the experiment is the tuning of the controller (figure 3.).

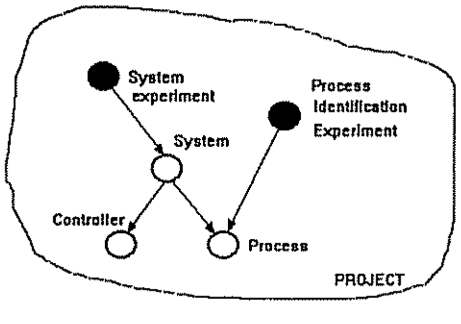

Figure 3. SIMBIOS hierarchical approach.

If as a result of the identification experiment, the process parameters are modified. The changes will be automatically seen when executing the system experiment.

The following subsections introduce the project, experiment, model and submodel SIMBIOS concepts.

Project

The project is responsible of the simulation project management. A project is composed of a set of submodels, models and experiments. The submodels are grouped into models and every model has one or more experiments associated to it. A submodel can be simultaneously present in more than one model and any change made to the parameters of that submodel will be automatically seen in all the models where the submodel is present. This capability must not be confused with the parameterized MACRO or the submodel block since they are templates (classes in object-oriented programming terminology) from which submodels can be instantiated. Therefore, in the present simulation languages several models can have instances of a particular submodel block. However, a particular instance is only active (visible) in a single model but not simultaneously in several models.

Figure 1 shows a model which is part of the *TurbineRegulation* project. This project has the following submodel set: *Limiter, RealPole, gate, error, motor, sigma, kt, Limiter, DeadSpace, MesureAmp, error1, Pulse, SinWave, Pulse1, ISE, SE, E* and *DataStream*. These submodels are grouped into the following models: *AutomaticModel* (figure 1.), *PrimaryRegulator* (figure 4.) and *NonLinearBlocks* (figure 5.). Notice that, for example, the *RealPole* submodel appears in two models, the *AutomaticModel* and the *PrimaryRegulator*. Thus, if as a consequence of an optimization study performed to the *PrimaryRegulator* model the *RealPole* time constant is set to a value that minimizes the Integral Square Error criteria (*ISE*), the *AutomaticModel* will also be affected since they both use the same submodel instance.

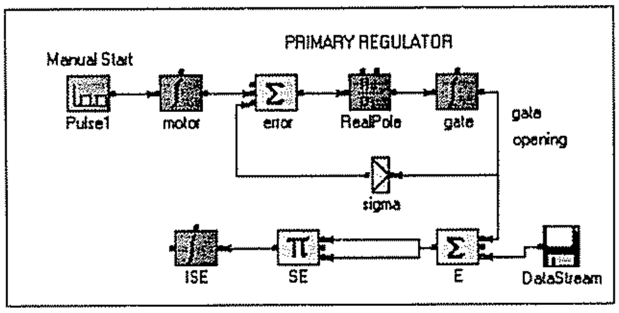

Figure 4. *PrimaryRegulator* model.

Experiment

In SIMBIOS and from the implementation point of view, we rather distinguish between a simulation experiment and a simulation study. A simulation experiment can be defined as a *run of a model and the associated experiment over a period of time from a known initial frame* and a simulation study can be defined as *a set of related simulation experiments* (Symons 1986).

The system currently has two predefined studies, the *sensitivity* study and the *batch* study:

Batch study: A set of experiments or studies placed on an ordered queue which are executed in its sorted order. For example, the first element of the queue can be an optimization study over the *PrimaryRegulator* that identifies the RealPole time constant, the second element can be an experiment over the *AutomaticModel* and the last element can be a sensitivity study which analyzes the sensitivity of the *PrimaryRegulator* response to parameter deviations.

Sensitivity study: A parameter is changed linearly from a start value to an end value. For each value of the parameter an experiment or study is execute.

Two more studies will be implemented, the optimization study and the random study.

The experiment does not allow the inclusion of dynamics external to the system model yet. They have to be included in the model.

Model

In continuous model simulation, a model or a submodel is described by a set of first-order differential equations written in terms of the state variables $(x_1,x_2, . . . ,x_n)$, the input variables $(u_1,u_2, . . . ,u_m)$ and the output variables $(o_1,o_2, . . . ,o_r)$. The continuous model takes the form:

$$x'(t) = f_1(t,x(t),u(t)) \qquad \textit{derivative computations}$$
$$o(t) = f_2(t,x(t),u(t)) \qquad \textit{output computations}$$
$$x(t_0) = f3(u(t_0)) \qquad \textit{initializations}$$

the model may include a set of discontinuity (constraint) functions:

$$0 = f_4(t,x(t),u(t))$$

The model discontinuities can be written in terms of the discontinuity functions $(g_1,g_2, . . . ,g_k)$:

$$g(t) = f4(t,x(t),u(t)) \qquad \textit{discontinuity functions}$$

A discontinuity is detected by noticing a change of sign in the value of a discontinuity function.

A model or a submodel includes, in general, code for its initializations, derivative computations, discontinuity function computations and output computations. Therefore, it seems advisable to cluster the code in sets, called segments in SIMBIOS, consistent with the previous division (Guasch 1987). Furthermore, *each segment can be directly implemented as a method* in the object oriented language.

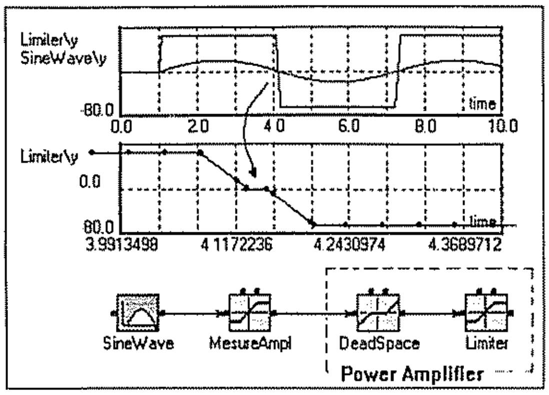

Figure 5. *NonLinearBlocks* model.

Figure 5. show the NonLinearBlocks model and its associated experiment results. Most of the discontinuous SIMBIOS submodels are accurately modeled using time and state event mechanisms.

Submodels

Since a primitive element, i.e. integrator, is a hand-made submodel, we also use in this section the term submodel to refer the lower level submodels (primitive elements).

SIMBIOS has a large set of linear, non-linear, algebraic, interface, discrete, fuzzy and event-driven submodels
The present submodel library is equivalent to that of available CSSL or block oriented continuous simulation languages. However, since submodels are implemented as objects, greater flexibility can be achieved.

Conclusions

This paper introduces the project concept implemented in the SIMBIOS object oriented simulation environment prototype. The objective of the project is to integrate in the simulation environment all the documentation, submodels, models and experiments that have been defined por a specific simulation project.

Smalltalk has been the language of choice for fast prototyping of SIMBIOS, but we are considering to port it to a more efficient language in run time such as C++. The migration of the SIMBIOS environment from a Smalltalk platform to a more performing one in real-time such as C++ is envisaged as feasible with a relative low effort.

SIMBIOS is available for distribution from the authors. The present prototype is currently available for PC 386 and 486 platforms.

References

Andersson, M. (1989),"Omola - An Object-Oriented Modeling Language", *Report TFRT-7417*, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

Beck B. (1990), "Shared-Memory Parallel Programming in C++", *IEEE Software*, pp. 38-59. (July).

Birtwistle G.M., Dahl B., Myrhaug B. and Nygaard K. (1973). SIMULA begin. Studentlitteratur.

Cellier F.E. (1991a),*"Continuous System Modeling"*, Springer-Verlag New York, Inc.

Cellier F.E, Zeigler B.P, and Cutler A.H. (1991b), "Object-Oriented Modeling: Tools and Techniques for Capturing Properties of Physical Systems in Computer Code," *Proceedings CADCS'91 -- IFAC Symposium on Computer-Aided Design in Control Systems*, Swansea, Wales, U.K., July 15-17, pp. 1-10.

Cellier F.E. and Elmqvist H. (1992), "The Need for Automated Formula Manipulation in Object-Oriented Continuous-System Modeling," *Proceedings CACSD'92 -- IEEE Computer-Aided Control System Design Conference*, Napa, CA, March 17-19 (1992).

Guasch A. (1987),*"MUSS: A Contribution to the Structural Analysis of Continuous System Simulation"*, Ph.D. Thesis, Universitat Politècnica de Catalunya, Barcelona, Spain.

Guasch A. (1991), "Continuous Simulation in Smalltalk", *1991 European Simulation Multiconference*, Copenhagen, Denmark, pp. 120-126, (June).

Helsgaun K. (1980) "DISCO - a SIMULA-based language for continuous, combined and discrete simulation", *Simulation*, Vol. 35 (1), pp. 1-12 (July).

Kreutzer W. (1986), *System Simulation: Programming Styles and Languages*, Addison-Wesley. ISBN 0-201-12914-0.

Mattsson S. E. (1990),"A Kernel for System Representation", *IFAC 11th World Congress*, Vol. 10, pp. 91-96.

Sim R. (1975), *CADSIM Users' Guide and Reference Manual*. London: Imperial College Publication #75/23.

Symons A. (1986), "Summary of some current issues", *TC3-IMACS Simulation Software Committee Newsletter No 86/01* (January).