In: Vichnevetsky, R., Borne, P. & Vignes, J. (eds.). Proc. of the 12th IMACS World Congress on Scientific Computation, July 18-22 1988, Paris, France, 5 Vols., 2: 761-763.

SAM - SIMULATION AND MODELLING SOFTWARE FOR WORKING STATIONS BASED ON MODELLING THEORY

Klara Vancso-Polacsek, Andreas Fischlin and Walter Schaufelberger

Projekt-Zentrum IDA
Swiss Federal Institute of Technology, CH-8092 Zurich, Switzerland

Abstract: This paper presents a new interactive modelling and simulation software *SAM* (**S**imulation **A**nd **M**odelling) written in Modula-2. *SAM* is based on modelling and simulation theory and was designed to run on working stations supporting a graphically oriented user interface with bit-map graphics, menu and window technique. A prototype of *SAM* was realized as a set of Modula-2 modules providing all functions required to model and conduct simulation experiments on systems of the classes Sequential Machine (SeqMach), Differential Equation System Specification (DESS), and Discrete Event System Specification (DEVS). The way, how the basic formalisms can be expressed in Modula-2, is shown using the formalism Sequential Machine. The modelling and simulation process with *SAM* is discussed in detail. The general structure of *SAM*, and its connection to the models is illustrated.

1. Introduction

The requirement of interactive modelling and simulation on personal computers and working stations requires modifications of traditional simulation software architectures. First, most currently available simulation software is not apt to be ported onto working stations without major modifications because it does not properly support a modern man-machine interface typical for advanced working stations. Furthermore, a traditional simulation software is usually not based on modelling theory. This paper presents a new interactive modelling and simulation software, *SAM* (Simulation And Modelling) which was designed to run on working stations and is based on modelling theory. It is implemented in Modula-2.
Working with an interactive program or software, the man-machine dialogue is dominant. The high-resolution graphics display of a working station supports a user-friendly man-machine interface with window technique, pull-down or pop-up menus. On the basis of these primary elements, a unified user interface can be defined which is implementable on different working stations. Simulation software built on top of such a unified interface can be made machine independent. Moreover, a high-resolution graphics display is useful for the graphical input and output of the modelling and simulation software.
The theory of modelling and simulation (1, 2, 3) became an independent discipline in the last years. Its role in the practical modelling and simulation is similar to the role of mathematics in the sciences and in engineering. In the past modelling and simulation have developed their own concepts of model description, simplification and validation, which are not specific to any particular discipline, but can be generalized. The theory guarantees a compact, homogeneous formalism not only for different models of the same class but also for different classes. Further, it makes modelling possible on various kinds of levels with the same formalism. If the theoretical concepts can be expressed in Modula-2 in a way, that they do not lose their abstract properties, then very different systems can be handled with the same software. This simplifies the implementation of a modelling and simulation software for a large class of systems substantially.
SAM is written in the high-level programming language Modula-2 (4), which was developed for constructing complex, modular software systems. One can take advantage of the benefits of Modula-2 on two levels: during the development of *SAM* and during the manipulation of models with *SAM*. To develop *SAM*, the programming language must be suitable for the programming of a user-friendly man-machine interface on working stations. The suitability of Modula-2 for such tasks was shown e.g. in (5). A further argument is, that efficient and inexpensive Modula-2 compilers were implemented on a large number of working stations in the last years. The modularisation supports the hierarchic, modular organization of models. Modula-2 allows the definition of structured data types. The implementation of abstract formalisms which are fundamental in the modelling theory requires powerful abstraction mechanisms in the programming language used. Modula-2 possesses such a mechanism, namely the so-called opaque types. The opaque types and the powerful facility of the procedure types were both needed in order to realize a proper design.

2. Definition of the Theoretical Concepts in MODULA-2

The transfer of the theoretical concepts in Modula-2 cannot be fully described here. Instead we illustrate the principle by transferring one of the three basic formalisms, the Sequential Machine Formalism. The Sequential Machine is given by a five-tuple in the form SeqMach=(X,Q,Y,δ,λ), where X, Q, Y, are the sets of the inputs, states and outputs. δ is the single step (or state transition) function, (i.e. δ: $Q \times X \rightarrow Q$) and λ is the output function (λ: $Q \times X \rightarrow Y$). Given the current state and input, δ

determines the next state of the system. The type SeqMach in Modula-2 can be defined within a single definition module in the following way:

```
TYPE
  Input; State; Output;                      (* X, Q, Y *)
  SingleStepFunction=PROCEDURE(State,Input):State;
                                                   (* δ *)
 OutputFunction=PROCEDURE(State,Input):Output; (* λ *)
 SeqMach = RECORD
             Delta: SingleStepFunction;
             Lambda: OutputFunction;
           END;
```

The sets of inputs, states and outputs are declared as opaque types. That means, that the modeller must define the corresponding data structures depending on the current model in the implementation module. The type SeqMach is given by a record structure, where the two components are function procedures with predefined formal parameters and result´s type. These declarations are suitable for all systems which can be interpreted as a SeqMach. The transfer of the three basic formalisms (SeqMach, DESS, DEVS) in Modula-2 was discussed in more detail in ref. (6).

3. Modelling and Simulation with *SAM*

The manipulations during the interactive modelling and simulation with *SAM* can be classified into two groups. The general manipulations, such as choosing the model or the experiment, can be executed for all types of models. The model specific operations depend on the model. Such a manipulation is e.g. the definition of a transition function for a current model of the type Sequential Machine. The general manipulations can be supplied by *SAM*, while the model specific ones must be written by the modeller in Modula-2. The formal parameter list of the procedures is predefined.
Figure 1 shows the architecture of *SAM*. The models "Model 1"..."Model n" are formulated as Modula-2 modules. They are connected with *SAM* via a predefined interface. The models are expressed by the corresponding procedures in the particular implementation modules. However, the structure of these modules must be known to *SAM*. For this reason the modeller must write the implementation modules for predefined definition modules only. There are three types of predefined definition modules: "SeqMach", "DESS" and "DEVS". They correspond to the abstract system formalisms defined in (1,2).
The modelling process with *SAM* is the following. The modeller edits the models of a certain class in an implementation module. The models are expressed by the model specific procedures. This requires the implementation of four procedures for a model of the type SeqMach: two procedures for the definition of the model itself (the transition function and the output function), one for the input needed by the current experiment and one to produce the display (or file) output of the simulation results. The procedures for the experiment input and the display output may be the same for different models, as long as they belong to the same class. On the other hand,

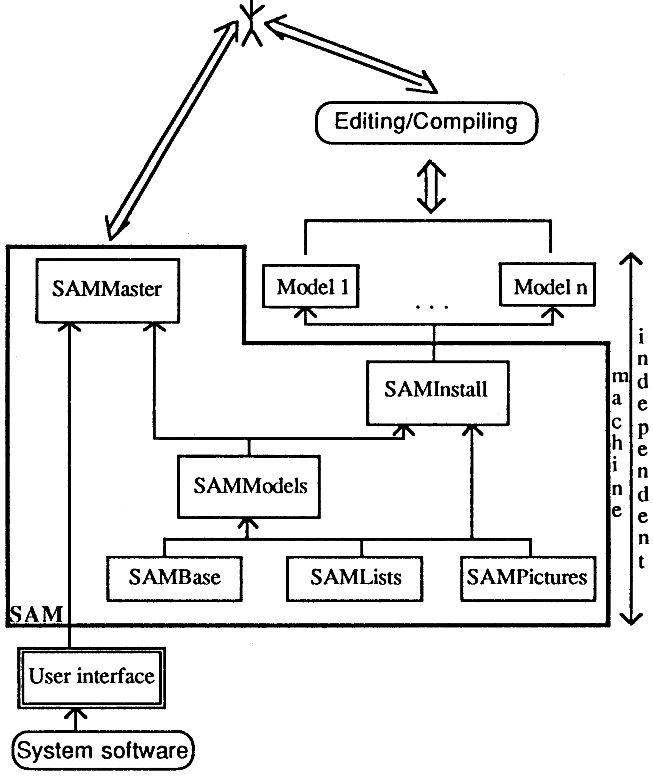

Figure 1: The general architecture of *SAM*. Any number and classes of models are provided by the modeller and are to be linked dynamically to *SAM* via predefined interfaces. Boxes correspond to Modula-2 modules, arrows signify imports.

there can also be different experiments and display procedures used for the same model.
Models are made ready by compiling their implementation module and installing them by means of *SAM*. Afterwards, the models may be used for interactive simulation. The modeller has the possibility to install new models or to delete the unused models at all times.
To reduce the programming work, the procedures, which are used frequently, can be collected into a separate library module. Once written the modeller can import them any time in order to define the current model. In this way the modeller can build a modelling environment specifically tailored to his individual needs. The data structures for the models in the class DESS (Differential Equations) are the same for all models of that class. Hence, for this class of models, even the so-called model specific procedures can be supplied by *SAM*.

4. The Architecture of *SAM*

SAM was designed in a modular fashion. Its architecture is shown in Figure 1. *SAM* consists of the modules within the bold frame. Single boxes correspond to Modula-2 modules; the arrows represent imports.
The module SAMMaster is the main program module. It controls the main activities within *SAM* and it is respon-

sible for the dialogue with the user. All other modules consist of a definition and an implementation part.
The interface between *SAM* and the models to be defined by the modeller consists of a few procedures. They can be imported from the module SAMInstall.
The modules SAMModels, SAMBase, SAMLists, SAMPictures are internal modules of *SAM*, and they export no objects of interest to the modeller.
The double border box "User interface" stands for the set of modules called "Dialog Machine" (5). They export objects necessary for a user-friendly man-machine interface and isolate *SAM* from the machine dependent system software required by the user interface. Not *SAM*, only the implementation parts of the "Dialog Machine" modules are system dependent.
The first prototype of *SAM* was implemented using the MacMETH Modula-2 Language System (7) on an Apple Macintosh II computer. Figure 2 shows a part of the screen during a Monte-Carlo simulation of a polymerization process modelled as a SeqMach.

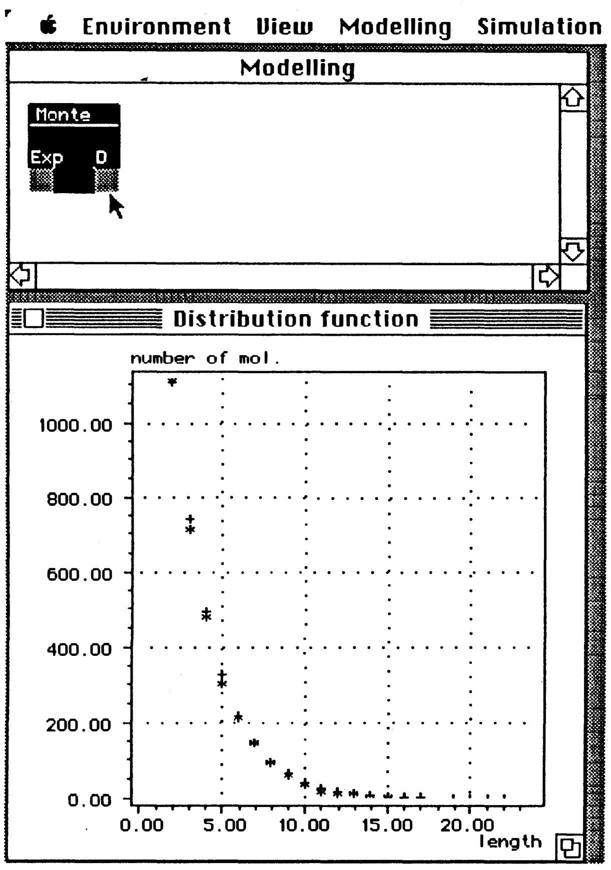

Figure 2: A part of the screen during a Monte-Carlo simulation of a polymerization process modelled as a SeqMach. The window with the title "Distribution function" presents the results of an experiment.

5. Discussion

The first results of our work are the definition of the concepts of modelling and simulation theory in Modula-2, and the design and development of a prototype of a new interactive modelling and simulation software, *SAM*.
The abstraction mechanism of Modula-2 was powerful enough to define the three basic formalisms (SeqMach, DESS, DEVS) in an abstract, compact form. The most important tools were the opaque and procedure types.
SAM does not offer a new simulation language, the models must be defined using the programming language Modula-2. The modeller has all the possibilities of this powerful, high-level programming language. However, the formulation of the models is restricted and generalized by formal definitions, which are based on the modelling and simulation theory. By means of these formal definitions it is possible to handle the models in the same class in a unified way, e.g. by *SAM*. Structured, hierarchical systems can be defined using the structured data types and the module concept of Modula-2.
The use of the modelling theory made it possible to build a simulation software, which is suitable to model and simulate very different classes of systems.
The adopted techniques have proved to be useful and flexible. For instance, *SAM*'s architecture allows the modeller to construct a "personal" modelling and simulation environment for frequently used models.
The prototype of *SAM* was implemented on the Apple Macintosh II. It has been found that such a high-resolution graphics display is fundamental not only for the user-friendly man-machine interface, but also for the flexibility of the experiment input and for the quality of the graphical output produced during simulations.
The user interface of *SAM* was based on the "Dialog Machine" which consists of a set of modules providing the basic interface objects of a modern working station (such as graphic windows, pull-down menus). The "Dialog Machine" has allowed to implement *SAM* in a simple and efficient way. Moreover, it has substantially supported the portability of *SAM*.

References

(1) ZEIGLER, B. P., *Theory of Modelling and Simulation,* John Wiley & Sons, 1976
(2) ZEIGLER, B. P., *System Theoretic Foundations of Modelling and Simulation,* in: ÖREN, T. I., ZEIGLER, B. P., ELZAS, M. S.(EDS): *Simulation and Model-Based Methodologies: An Integrative View,* Springer-Verlag, 1984
(3) WYMORE, A.W., *Theory of Systems* in: VICK, C. R., RAMAMOORTHY, C. V.(EDS.): *Handbook of Software Engineering,* Van Nostrand Reinhold Company, New York, 1984
(4) WIRTH, N., *Programming in Modula-2, Third, Corrected Edition,* Springer-Verlag, 1985
(5) FISCHLIN, A., *Simplifying the usage and programming of modern working stations with Modula-2: "The Dialog Machine",* Project-Centre IDA, ETH Zürich, 1986.
(6) Vancso, K., Fischlin, A., Schaufelberger, W., *The Development of Interactive Modelling and Simulationssoftware with Modula-2,* (in German), in: Halin, J.(ed.):, "Simulationstechnik", 4. Symposium Simulationstechnik, Zürich, 9.-11. September 1987, Proceedings, Springer-Verlag, Informatik-Fachberichte 150, pp. 239-249.
(7) Wirth, N., Gutknecht, J., Heiz, W., Schär, H., Seiler, H., Vetterli, Ch., *MacMETH: A Fast Modula-2 Language System for the Apple Macintosh,* Institut für Informatik, ETH Zürich, 1986.