*Proceedings of the 1996 Winter Simulation Conference*
*ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain*

# CSIM18 - THE SIMULATION ENGINE

Herb Schwetman

Mesquite Software, Inc.
3925 West Braker Lane
Austin, TX 78759-5321, USA

## ABSTRACT

A simulation engine is the collection of components, features and support functions which are crucial to the implementation of an efficient discrete event simulation model. Furthermore, this model can be embedded in a larger application. A good simulation engine combines efficiency, functionality and completeness to enable a model builder to construct a comprehensive, customized model of either a specific system or a class of systems. This paper presents CSIM18, a simulation engine which supports development of applications with efficient, embedded simulation models on a variety of system platforms.

## 1 INTRODUCTION

Many application developers are finding a need to embed a simulation model into their application. Applications which help users design or configure complex systems for which performance is a major design goal are examples of such applications. This paper presents CSIM18, a simulation engine which has the features that developers find useful as they add simulation models to their applications.

An example of such an application would be a database design and layout program. In many cases, a database management system (DBMS) is used with a high performance on-line transaction processing (OLTP) system. A major goal of the system is to give "quick" responses to each kind of transaction in the stream of incoming transactions. The design program then needs to help the designer layout the database so that the response time requirement can be met. In order to evaluate a design, the designer needs to obtain estimates of transaction response times for the design just completed. A simulation model of the OLTP system with the DBMS as specified in the design program could provide the necessary estimates of the response times for each kind of transaction as the operation of the system is simulated in a realistic fashion. In this example, the design program is the larger application, and the model of the OLTP system and the DBMS is the embedded simulation model. In many cases, it would be appropriate and expedient to implement the embedded model with a simulation engine.

As this application is being designed and implemented, and the need for a simulation engine determined, a natural question is the one of "make or buy" - should the project implement their own simulation engine or should they examine the software marketplace, to see if an appropriate simulation engine can be purchased? Given the economics of software development, it is usually advantageous to buy a software component, if one can be found that (1) fills the need, (2) is compatible with the other software tools and components being used, (3) is supported and (4) is "reasonably" priced.

This paper first describes a simulation engine and presents the features which distinguish a good simulation engine. It next presents CSIM18 and shows why it fulfills the need for a simulation engine in many applications. The paper concludes with some examples of applications which have incorporated CSIM18 (or its predecessors) as the simulation engine for their models.

## 2 SIMULATION ENGINES

A simulation engine is a set of objects and methods which are used to construct simulation models which are embedded in applications. This is in contrast to a simulation environment which is itself the complete application. In many cases, an application is tailored to specific domain. In fact, an application can be quite complex, and the embedded simulation model is really a just minor component, usually used to help evaluate some aspect of the application.

The key point is that the simulation engine is used to build a model which is embedded in a larger application. The model is not the goal; the model is used to support the goal (the complete application). Some examples of applications with embedded models will be given in a

later section.

There are several features which characterize a "good" simulation engine; these include:

a. the engine is a *complete* simulation package; it can support all of the important discrete-event simulation paradigms, including the process-oriented paradigm;
b. a model implemented with the engine can be an *accurate representations* of the "real" system; in particular, important features of the real system, including complex component interrelationships, unusual scheduling rules, and varying workload profiles can all be represented in the model;
c. models of almost any *size* can be created with the engine;
d. the engine is *efficient*; models constructed with the engine execute in an "reasonable" amount of time; in addition, both large and small models execute efficiently;
e. models based on the engine are *compact*;
f. models based on the engine are written in *standard programming languages*; in particular, developers do not have to learn a new language which is peculiar to the simulation model;
g. the model is *embeddable*; the model can be easily integrated with the rest of the application;
h. the engine has an *easy-to-use* interface; and
i. the model can be retargeted to *multiple platforms*; the model should not be an impediment to moving the application to different systems.

There are some additional features which could simplify the task of the application developer. These include:

a. *Automatic run-length control*: methods for determining when accurate results are achieved
b. *Random number streams* and *random number functions*: routines for generating streams of varying values from many different probability distributions, and
c. *Data collection* and *reporting routines*: routines for collecting data and generating useful ports.

When an application developer is evaluating simulation engines, these and other factors will be used to help select the best engine for use in the project.

It should be noted that a graphical user interface (GUI) was not listed as an important feature for a simulation engine. It is assumed that the larger application has its own GUI and that a GUI specific to the simulation model is inappropriate and not needed.

## 3 CSIM18

CSIM18 is a complete simulation engine. Earlier versions of CSIM18 are described in a number of articles (Schwetman 1986, 1988, 1990, 1994, 1995). These predecessors to CSIM18 have been used to model a large variety of systems, including communications systems [Edwa92], computer systems, transportation systems, microprocessors, and fault tolerant systems.

CSIM18 is a library of classes, functions, procedures and header files which enable developers to implement efficient models of complex systems. The models are written in either the C or C++ programming languages. The library is available on many different system platforms, including PC's with either Windows 3.1, Windows 95, Windows NT, OS/2 Warp and Linux operating systems. It is also available on almost all UNIX workstations, including Sun SPARC (SunOS and Solaris), DEC Alpha (with OSF/1), HP PA (with HP/UX), IBM RS/6000 (with AIX) and SGI workstations. There is also a version for the Power Mac (with the Metroworks C++ compiler).

CSIM18 supports process-oriented, discrete-event simulation models. In this kind of model, the complete system is represented by a collection of simulated resources and a collection of processes which compete for use of these resources. Typically, such a model is used to provide estimates of the performance of the real (modeled) system. The goal is to use the model to "try out" different system configurations or different workloads or different resource scheduling rules, to find the "best" configuration (or workload or schedule) with respect to the performance goal. In most cases, it is much "cheaper" to try out configurations in a system model than it is to use the real system.

The specific classes (simulation components) provided with CSIM18 (Mesquite 1996) include:

a. Processes: a CSIM process is an independent thread of execution; the active entities of a model are implemented as processes; since many processes can appear to be executing simultaneously, processes can mimic the activities of entities operating in parallel;
b. Facilities (these are called "active" resources in some simulation systems) - resources are typically "used" by processes in the model; facilities have one or more servers; processes queue up for access to a server;
c. Storages (these are called "passive" resources in some simulation systems) - a storage is initialized with a number of storage units; processes can "allocate" some subset of the storage units;
d. Events - events are used to synchronize actions of different processes, and
e. Mailboxes - mailboxes are used to interchange information between processes.

In addition, CSIM18 provides classes (simulation components) to assist the with collection of data in a model; these include:

a. Tables - used to collect real valued (double) data values and to report on their statistical properties;
b. Qtables - used to collect integer valued, time depend-

ent data values (e.g. queue lengths) and to report on their statistical properties;

c. Meters - used to measure the flow of entities past a point in the system and, also, to measure the times between successive passages of entities past this point; and

d. Boxes - used to collect data on time spent in a specified range of process activities.

A standard report can be generated for each facility, storage, table, qtable, meter and box. Histograms and confidence intervals can be requested for each table, qtable, meter and box. In addition, "inspector" functions are provided to give programs access to every data item collected for all of the structures with report capabilities. The developer can use these inspector functions to produce reports tailored for the specific application.

A "run-length-control" feature can be associated with the mean value of a table, qtable, meter or box (Mesquite 1995). This feature allows the modeler to specify a "stopping rule" for execution of the model. The stopping rule is based on achieving a specified level of confidence for some output statistic. A CPU time limit is also specified, so that the model will cease execution (with possibly inaccurate results) when this time limit is exceeded. This run-length-control feature could be very important for applications which depend on achieving accurate results in an efficient manner from the embedded simulation model.

CSIM18 has some other features which will assist the implementation of embedded models. These include:

a. the "rerun" statement: used to "tear down" a model, so that a new model can be constructed by the program;

b. the "reset" statement: used to discard results collected so far from all of the data collection facilities in a model; this used to allow for a "warm up" interval before actual data collection is started; and

c. model debugging aids: a selective event trace can be used to give details on selected (or all) aspects of the execution of the model; for some platforms, an interactive execution monitor lets the developer examine, interactively, many parts of the model and also lets the developer "step through" execution of the model.

Models produced with CSIM18 can execute very efficiently. A test program has been written and executed on many different platforms. The test consists of a model of an M/M/1 queue with 50% server utilization operating for 5000 arrivals and 5000 departures. The model requires 17,485 simulation events. The following tables shows the timings for executing this model on several platforms:

| System: | Dell PC | SPARC | RS/6000 |
|---|---|---|---|
| Processor: | Pentium 133 | SStation5 | PowPC 601 |
| OS: | Win95 | Solaris | AIX |
| Compiler: | MS VC++ 4.0 | g++ 2.7 | xlc xx |
| User CPU sec: | 0.430 | 1.580 | 0.450 |
| Events per sec: | 40,662 | 11,066 | 38,856 |

CSIM18 is delivered with a complete set of documentation consisting of a Users' Guide (for either the C or C++ version) and a introductory ("Getting Started") manual, along with installation instructions.

CSIM18 is the newest version of a series of CSIM simulation software toolkits. CSIM was originally developed in 1985 at MCC. The preceding version, CSIM17, was developed and sold by Mesquite Software, Inc. (under the terms of a license agreement with MCC) in 1994. CSIM18 incorporates suggestions, enhancements and improvements from many sources including customers and developers using CSIM17.

## 4 EXAMPLES

Several projects have used previous versions of CSIM18 as a simulation engine for embedded models. This section describes some of these projects.

### 4.1 VisSim/Discrete Event

Visual Solutions, Inc., sells VisSim (Visual Solutions 1995), a simulation package for developing continuous, discrete, multi-rate and hybrid system models and for performing dynamic simulations. VisSim incorporates an outstanding graphics user interface which is used to create diagrams of the systems being simulated. Each model consists of a collection of components connected by flexWires. Components can perform many kinds of activities such as evaluating a mathematical function, generating a random value, performing an arithmetic or logical test, and producing different forms of animation. VisSim add-ons allow users to find optimal solutions to mathematical programming problems, model fuzzy-logic systems and do real-time control functions.

Some of the VisSim users expressed a need for discrete-event simulation capabilities in the VisSim modeling building environment. Visual Solutions responded to this need by engaging Mesquite Software to design and implement VisSim/Discrete-Event (Visual Solutions 1996), an add-on to VisSim which lets modelers use the VisSim GUI to construct, test and execute discrete-event simulation models. CSIM17 (the predecessor to CSIM18) is the simulation engine which "powers" VisSim/Discrete-Event.

Briefly, new VisSim blocks which represent most of the CSIM classes and methods are added to VisSim. These blocks are interconnected by VisSim flexWires. The model consists of one or more of "process diagrams". Process diagrams represent creating processes

(the source block), terminating processes (the sink block) and many different kinds of process activity blocks (such as useServiceStation, allocateStorage, waitEvent and computeResult). Other blocks represent serviceStations (active resources), storages (passive resources), events and tables. Some of the standard VisSim blocks are used to present output results (display blocks, meters and strip charts) and perform some kinds of arithmetic operations. Compound blocks, which are aggregations of blocks, can be used to represent submodels.

VisSim/Discrete-Event is a powerful tool for creating discrete-event simulation models with a block diagramming interface. The models can represent many styles of process behavior and can be used to model many different kinds of systems. The internal simulation engine, CSIM17, when combined with the VisSim GUI enabled this new modeling environment to be created with a reasonably modest effort (about 2/3 of a man-year).

### 4.2 ArchGen

CAE + Corporation has developed and is selling a CAD system called ArchGen (Jain 1995, 1996). ArchGen captures, validates and optimizes system IC specifications. It includes a CSIM model for functional and performance simulation with clock edge accuracy. CSIM17 is the simulation engine used in ArchGen.

CAE+ decided to use CSIM because they needed a model which could be "connected" to the rest of the ArchGen components, and they needed a model which has a high rate of executing basic simulation activities (the IC designer is usually sitting at the workstation, waiting for the model to produce its estimates of chip performance). Mesquite Software has worked closely with CAE+, to insure that the functionality and performance needed by ArchGen would be available in CSIM.

### 4.3 *SPE•ED*

L&S Computer Technology, Inc. sells *SPE•ED* (Smith and Wong 1994), a tool for modeling, at a high level, the functionality and behavior of large software systems and for providing estimates of the performance of these systems in operation. *SPE•ED* supports the methods and techniques of performance engineering, to help software designers translate their designs to efficient implementations. L&S Computer Technology selected CSIM as the simulation engine because it was a "excellent fit" with the needs and capabilities of *SPE•ED*. In addition, CSIM was available on the platforms that L&S needed

### 4.4 Others

A large computer manufacturer introduced a large scale multiprocessor system and a parallel relational database management system (RDBMS). The system supports both high-volume on-line transaction processing and large, complex decision support queries. The manufacturer decide to offer a configuration planning and management application, to support both layout of the database and performance tuning of the transaction processing software. This application incorporated a simulation model of the multiprocessor system along with the parallel RDBMS and the transaction processing software. CSIM was chosen to be the basis for the simulation model. The key reasons for selecting CSIM were that CSIM could be easily integrated into the rest of the application, CSIM provided the necessary functionality for implementing the model, and CSIM provide efficient execution of the model. The efficient execution of large models turned out to be important, because some the configurations suggested for customers were very large and the transactions being simulated could be quite complex.

A research project at a large university built a software system for designing and evaluating fault-tolerant computer systems. It used CSIM as the simulation engine for the models of failures and repairs in the systems being designed.

A consulting firm built a model of inventory management system for a warehousing operation. They used CSIM as the basis of the model used to "try out" different management rules. The application was written in Visual Basic for execution on a PC with Windows. The Visual Basic application connected to the model which was implemented as a DLL written in Visual C++ and the CSIM17 library.

## 5 SUMMARY

CSIM18 is a simulation engine. It can be used to implement simulation models which will be embedded in domain specific applications. Rapid development of customized applications has been greatly enhanced by the emergence of application development environments such as Microsoft Visual C++, Borland C++, Microsoft Visual Basic and other similar products. One project has recently developed an application based on an Excel spreadsheet, with a link to a CSIM model.

The predecessor of CSIM18, CSIM17, has been successfully used to develop realistic models of many different kinds of complex systems. Some examples of applications incorporating CSIM models have be given in this paper. CSIM18 offers several new features which

will be essential to developers. The functionality, compactness and efficiency of CSIM18 mean that models based on CSIM18 will enhance the surrounding applications.

## ACKNOWLEDGMENTS

CSIM is copyrighted by Microelectronics and Computer Technology Corporation (MCC). CSIM17 and CSIM18 are supported and marketed by Mesquite Software, Inc. under a license from MCC. Dr. Jeff Brumfield developed the new data collection and presentation functions, including the run length control algorithm, in CSIM18.

## REFERENCES

Edwards, G. and R. Sankar. 1992. Modeling and simulation of networks using CSIM. *Simulation* 58(2): 131 - 136.

Jain, P. 1995. A comprehensive pre-RTL design methodology. In *Proceedings of the 1995 International Verilog HDL Conference*. IEEE Computer Society Press.

Jain. P. 1996. Specification validation of embedded system ICs. In *Proceedings of Embedded Systems Conference 1996*. Miller Freeman, Inc.

Mesquite Software, Inc. 1995. Confidence intervals and run length control in CSIM18. Internal Report. Austin, TX.

Mesquite Software, Inc. 1996. *CSIM18 Users Guide*. Austin, TX.

Schwetman, H. 1986. CSIM: A C-based, process-oriented simulation language. In *Proceedings of the 1986 Winter Simulation Conference*, ed. J. Wilson, J. Henriksen, and S. Roberts, 387 - 396. Washington, DC.

Schwetman, H. 1988. Using CSIM to model complex systems. In *Proceedings of the 1988 Winter Simulation Conference*, ed. M. Abrams, P. Haigh, and J. Comfort, 246 - 253. San Diego, CA.

Schwetman, H. 1990. Introduction to process-oriented simulation and CSIM. In *Proceedings of the 1990 Winter Simulation Conference*, ed. O. Balci, R. Sadowski, and R. Nance, 154 - 157. New Orleans, LA.

Schwetman, H. 1994. CSIM17: A simulation model-building toolkit. In *Proceedings of the 1994 Winter Simulation Conference*, ed. J. Tew, S. Manivannan, D. Sadowski, A. Seila, 464 - 470. Orlando, FL

Schwetman, H. 1995. Object-oriented simulation modeling with C++/CSIM17. In *Proceedings of the 1995 Winter Simulation Conference*. ed. C. Alexopoulos, K. Kang, W. Lilegdon, D. Goldsman, 529 - 533. Washington, D.C.

Smith, C and B. Wong. 1994. SPE evaluation of a client/server application. In *Proceedings of Computer Measurement Group*. Orlando, FL.

Visual Solutions, Inc. 1995. *VisSim users' guide*. Westford, MA.

Visual Solutions, Inc. 1996. *VisSim/Discrete-Event user's guide*. Westford, MA.

## AUTHOR BIOGRAPHY

**HERB SCHWETMAN** is founder and president of Mesquite Software, Inc. Prior to founding Mesquite Software in 1994, he was a Senior Member of the Technical Staff at MCC from 1984 until 1994. From 1972 until 1984, he was a Professor of Computer Sciences at Purdue University. He received his Ph.D. in Computer Science from The University of Texas at Austin in 1970. He has been involved in research into system modeling and simulation as