Proceedings of the 1998 Winter Simulation Conference
D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, eds.

MODEL-BASED SYSTEMS ANALYSIS USING CSIM18

Herb Schwetman

Mesquite Software, Inc.
PO Box 26306
Austin, TX 78755-0306, U.S.A.

ABSTRACT

Achieving the best possible cost performance for a system is often the goal of the manager of that system. This is true for large systems (such as an entire railroad system) as well as for small systems (such as a computerized departmental transaction processing system). One technique for achieving this management goal is to develop and use an operational model of the system. This paper describes model-based systems analysis using simulation models of systems. This technique for finding and addressing performance related problems is widely used in a variety of industries. In many cases, the resulting improvements in cost-performance characteristics for the system can be significant.

1 INTRODUCTION

Managers of systems (both large and small) need to configure and manage their systems so that the systems deliver high levels of performance and incur the lowest possible cost. A number of factors make effective management of these systems difficult; these factors include:

- increasing size and complexity of systems
- changes in the demand for system services
- availability of more options in configuring systems
- demands for reducing system costs, and
- greater expectations for improved performance by customers.

One widely used technique for assessing and improving the cost performance characteristics of a system is to develop an operational model of this system and then use this model to “try out” different system configurations and different workloads. A common tool for developing such models is discrete-event simulation. This paper presents some details and examples of model-based systems analysis based on CSIM18, a tool kit for implementing process-oriented discrete-event simulation models (Schwetman, 1996).

A systems analysis project is usually initiated in one of the following types of situations:

- A new system is being designed and implemented (or acquired); there are questions about component trade-offs, and the best choices are often difficult to determine,
- An existing system is delivering unsatisfactory performance; in many cases, the unsatisfactory performance is evidenced by customer complaints, slow response times and low levels of service; in many cases, the result can be reduced revenues, higher operating costs and lost business.
- The workload for an existing system is predicted to change; the managers of the system need to provide the best possible estimates of the impacts on system performance that will result from these changes.

In all of these cases, there is a need to estimate system performance for systems and/or workloads which do not currently exist. Model-based systems analysis is the technique often used to address the need to predict performance in these situations.

2 METHODOLOGY

Model-based systems analysis typically proceeds in the following steps:

- Develop an operational model of the base-line system;

- For this model, develop a representation of the workload for this base-line system;

- Validate the accuracy of the system model processing the representation of the workload;

- Use the model to estimate performance with alternative configurations;

- Use the model to predict performance as the workload changes.

In this paper, a system is a collection of components which, collectively, process or service requests for service. The workload is a sequence of requests for service. Examples of systems which fit this description include:

- An assembly line assembling parts into finished items; each item is an element of the workload;

- A on-line computer system in a bank processing inquiries and transactions from bank tellers and customers; each inquiry or transaction is element of the workload;

- A large railroad processing customer car orders to pickup, transport and deliver batches of materials or goods is a system; the car orders are the elements of the workload;

- A central processing unit (CPU) in a computer processing a stream of instructions is a system; each instruction is an element of the workload for this system;

- A telephone call center responding to incoming calls from customers; the stream of incoming calls constitutes the workload for this system.

Model-based systems analysis is used to predict the performance of these kinds of systems when the actual system cannot be used to obtain estimates of performance.

3 PERFORMANCE

System performance is a measure of how a system is processing its workload. Typical measures of performance include:

- response time - the time total time required to process and complete a request for service; in many systems, the average response time is the single most important performance measure

- throughput rate - the rate at which the system completes requests for service;

- queue length - the number of requests for service in a system; in many cases, the queue length includes both requests waiting for service and requests receiving service.

Most systems consist of a collection of facilities or service stations or components. Many, but not all, of these facilities have one or more “servers” and a queue for waiting requests. Some facilities have multiple servers, and some facilities have no queue for waiting requests. The measures of performance listed above for systems can also be specified for these facilities. Thus, response time at a facility, the throughput rate at a facility and the queue length at a facility are performance measures for these facilities. In addition, the utilization of the server (the percent of time the server is busy or occupied) or servers at a facility is also used to characterize the performance of that facility.

Customers of a system are typically submitting requests for service from the system. A customer’s opinion about the performance of that system is often influenced by the response time for these requests. A system manger’s job often consists of balancing the customers’ desires for great service (reduced response times) with the system owner’s desire for low capital and operating costs. Unfortunately, many of these systems are very complex, and it is difficult to estimate or predict performance (response time) of such systems. The goal of model-based systems analysis is to assist in making these estimates and predictions. Still, it the managers responsibility to judge the quality of service (is response time OK or not) and to make the decisions to spend (or not spend) the funds required to improve or change the performance experienced by the customers of the system.

4 AN EXAMPLE

Telephone call centers are increasingly becoming the main link between a company and it customers. In many companies, customers order products through a call center, they obtain order status through a call center, they seek assistance in using products through a call center, and they deal with problems and other issues (such as warranties, etc.) through call centers. Customers’ view of the quality of the company can be heavily influenced by the service they experience as they deal with the company through these call centers. A major factor in forming these views can be the response times they experience as they access call centers. Moreover, as customers experience unacceptably long response times, they tend to “hang up” and leave as dissatisfied customers. Thus, a goal for the managers of such call centers could be to maximize the number or percentage of calls that successfully complete for each call center.

The response time for a call coming into a call center is influenced by many interrelated factors; these include:

- the number of incoming telephone lines,
- the capacity of the call switch, and
- the number of operators.

In addition, the rate that calls arrive at the center, the length of each interaction with an operator and the number of aborted calls also impact call response times. Balancing the need to offer acceptable levels of service at the call center with the costs of acquiring additional telephone lines, a larger call switch, and hiring and training additional operators makes the job of the manager of the center very difficult.

A simulation model of a typical call center can assist the manager of such a center in deciding how best to allocate funds for improving response times at the center. A CSIM18 version of such a model is described in the appendix to this paper. The focus of this section is on understanding the functionality of the model and the ways it can be used to make predictions about the operation of a telephone call center.

In the simplest terms, the model accepts a set of (input) parameter values and produces a set of (output) data values. The parameter values completely specify the quantities that determine the operation of the call center. For a test run, these were as shown in Table 1.

Table 1: Input Parameters

Number of telephone lines	10	
Number of switch ports	10	
Number of operators	10	
Mean operator service time	90.0	sec
Answer recording time	30.0	sec
Incoming call rate	0.10	calls/sec
Maximum caller wait time	120.0	sec

In the model, a new call tries to obtain a free incoming line. If there is no line, the call is rejected with a busy signal and leaves as an uncompleted call. A call with a line then contends for a port on the switch at the center. Such a call can either “time out” waiting for a port (the caller hangs up) or it can get a port and continue. A call with a port delays while the answer recording plays, and then contends for an available operator. Again, a waiting call can “time out” (the caller hangs up) and leave as an incomplete call or it can obtain an operator, delay while the caller communicates with the operator and leave a completed (successful) call.

The results from executing the CSIM18 simulation model of this call center are summarized in Table 2.

Table 2: Summary of Results

Call started	5007
Aborted calls busy	1555
Aborted calls port	0
Aborted calls operator	0
Call completed	3445
Calls in process	7
Fraction completed	0.69
Operator utilization	6.26
Port utilization	0.84
Line utilization	0.84

In this configuration, about 69% of the incoming calls are completed as successful calls. This set of results suggests that with more lines (and possibly more ports), the center could complete a higher percentage of the incoming calls.

A series of runs of this model with different configurations of lines, ports and operators demonstrates that the percentage of completed calls can be improved; these results are summarized in Table 3.

Table 3: Summary of Multiple Runs

Lines	**Ports**	**Operators**	**Fraction Completed**
10	10	10	.69
15	10	10	.77
15	15	10	.89
15	15	15	.91
20	15	15	.96

These results are examples of the kinds of use that can be made of this model by the managers of the call center. Assuming that the model can be validated as being capable of providing accurate estimates of these and other performance data, it (the model) can be used to accomplish many of the goals listed in the first section.

SUMMARY

Model-based systems analysis is an widely used technique for analyzing and improving the performance of many different kinds of systems. These systems are characterized by having a collection of resources and a set of requests (the workload) which need to “use” some or all of these resources. The performance of such a system is typically expressed in terms of the time required to complete these requests.

This paper has shown how to apply a simulation model as the basis for model-based systems analysis. The example, analyzing a telephone call center, has shown a set of improvements which can increase the percentage of completed calls.

ACKNOWLEDGEMENTS

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

APPENDIX

A CSIM18 model is a C or C++ program that implements a process-oriented, discrete-event simulation model of a system. The example in this paper is a C++ model of a telephone call center. The listing of the entire model is available on the Mesquite Software, Inc. web site (http://www.mesquite.com/). This appendix summarizes some of the main points of the model.

In the example, there are two C++ objects: a call object and a call_center object. The main procedure in the model is as shown in Figure A-1. It is designated the "sim" procedure and is a CSIM18 process.

```
extern "C" void sim()
{
   initModel();        // initialize model
   create("sim");      // create sim process
   initRun();          // initilize this run
   gen();              // start call generator process
   converged.wait();   // wait for valid results
   printResults();     // print results this run
   report();           // CSIM18 report
}
```

Figure A-1: Listing of Main (Sim) Process

This "sim" process initializes parts of the model; it then invokes the "gen" process and waits for the collected result (in this example, the percentage of started calls which complete successfully) to "converge" to a statistically valid (stable) quantity. The mechanism for doing this is based on the method of batch means (Law and Kelton 1991). The way this is implemented in CSIM18 is described in (Schwetman and Brumfield, 1997).

The "gen" process creates a new call object and then gives the call to the call_center object via the startCall method. The "gen" process then delays for the appropriate amount of simulated time and generates the next call. This process of generating a call and delaying for an inter-call interval continues until the model stop.

The startCall method is shown in Figure A-2. The callCenter object has three simulation objects which represent the pool of incoming lines (a CSIM18 storage object), the ports on the call switch (a CSIM18 storage object) and the operators (a CSIM18 multi-server facility) respectively. Each time the startCall method is called, it becomes a CSIM18 process by executing the "create" statement. Because each call is handled by a process, the model can have multiple calls "active" at the same time in the call center.

The processing of each call begins by checking for an available line: if there are no available lines, the call leaves immediately; if there is an available line, the call next tries to allocate a port on the switch. The "timed_allocate" method has a time-out interval specified. Thus, the call can either time-out or succeed. A successful call then delays while the recording plays ("hold(recordingTime);") and then uses the "timed_reserve" method to try to gain access to an operator. As each call leaves, it "records" a one in a CSIM18 table if it completed successfully or a zero in this table if it did not complete successfully. The average value computed by this table is the percentage of calls started that complete successfully.

The rest of the program consists of procedures to "setup" and initialize the model and to print the results.

REFERENCES

Law, A. and D. Kelton, 1991. *Simulation Modeling and Analysis*. (Second Edition) MacGraw-Hill.

Mesquite Software, Inc., 1997. *User's Guide, CSIM18 Simulation Engine*. Austin, TX.

Schwetman, H., 1996. CSIM18 - The Simulation Engine. In *Proceedings of the 1996 Winter Simulation Conference*. ed. J. Charnes, D. Morrice, D. Brunner, and J. Swain, 517 - 521. San Diego, CA.

Schwetman, H. and J. Brumbfield, 1997. Data Analysis and Automatic Run-Length Control in CSIM18. In *Proceedings of the 1997 Winter Simulation Conference*. Ed. S Andradottir, K. Healy, D. Withers, and B. Nelson, 687 - 692. Atlanta, GA.

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 applied to computer systems since 1968.

```
    void callCenter_c::startCall(call_c *call)
    {
      TIME x;
      long st;
      long result;

      create("call");
      result = 0;
      numCallsStarted++;
      numCallsInProcess++;
    // check line available
      if(lines->avail() > 0) {
        lines->allocate(1);
        x = uniform(0.0, call->getTotalAbortTime());
    // contend for switch port
        st = ports->timed_allocate(1, x);
        if(st != TIMED_OUT) {
    // listen to recording
          hold(recordingTime);
          x = call->getTotalAbortTime() - x;
    //contend for operator
          st = operators->timed_reserve(x);
          if(st != TIMED_OUT) {
    // converse with operator
            hold(uniform(operMin, operMax));
            result = 1;
            numCallsCompleted++;
            operators->release();
            ports->deallocate(1);
            lines->deallocate(1);
    // leave -  completed call
          } else {
            numCallsAbortOperator++;
            ports->deallocate(1);
            lines->deallocate(1);
    // leave - waiting for operator
          }
        } else {
          numCallsAbortPort++;
          lines->deallocate(1);
    // leave - waiting for port
        }
      } else {
        numCallsBusy++;
    // leave - busy signal
      }
      numCallsInProcess--;
      pcCompleted->record((double)result);
    // call leaves center
}
```

Figure A-2: Listing of Call Process