Proceedings of the 1997 Winter Simulation Conference
ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson

DATA ANALYSIS AND AUTOMATIC RUN-LENGTH CONTROL IN CSIM18

Herb Schwetman

Mesquite Software, Inc.
4210 Spicewood Springs Road, #201
Austin, TX 78759, U.S.A.

Jeff Brumfield

Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712, U.S.A.

ABSTRACT

The data collection and the automatic run-length control features provided in the CSIM18 library allow model builders to easily collect valid data from a simulation model and to be assured that statistically valid results have been achieved at a reasonable computational cost. This paper gives an overview of the CSIM18 library and then presents these features by using an example of a system model.

1 INTRODUCTION

CSIM18 is library of objects, methods, functions and procedures used by C/C++ programmers to construct simulation models of complex systems Mesquite Software, 1997]. CSIM18 provides the model builder with a comprehensive set of objects which simplifies the model building process. In addition to the objects and procedures used to implement the simulated resources and the processes (entities) of a model, other objects and procedures facilitate the collection of data in the model.

This paper describes these data collection facilities in CSIM18 and describes some of the features which help users obtain statistically accurate outputs from their models. These features mean that powerful models can be implemented, using CSIM18 as the underlying simulation engine [Schwetman, 1996], and then placed in the hands of inexperienced users with some assurances that these users will be able to make correct use of the results from the model.

2 CSIM18 OVERVIEW

A CSIM18 model consists of a collection of simulated resources and a collection of processes which interact with each other and compete for use of the simulated resources. The resource objects provided in CSIM18 are as follows:

- facilities,
- storages,
- mailboxes, and
- events.

In more detail, a *facility* has one or more servers and a queue for processes waiting to gain access to a server. Processes can either *reserve* a server for its exclusive use or *use* a server for a specified period of time. A process reserving a server will later *release* that server.

A *storage* is a pool of indistinguishable tokens or units of storage and a queue for tasks waiting to allocate a subset of the tokens. A process executes an *allocate* operation, specifying the number of tokens required from a storage. If there is not a sufficient number of tokens available to satisfy the allocate, then the process will wait until other processes have deallocated tokens that they have previously allocated. A process with tokens from a storage will eventually *deallocate* those tokens, returning them to the pool.

A *mailbox* is used to implement interprocess communications in a model. One process can *send* a message to a mailbox. Another process can do a *receive* operation on that mailbox; if there is a message waiting at the mailbox, then the process will get that message from the mailbox and continue. If a process does a *receive* and the mailbox is empty (no waiting messages), then the process will wait until another process sends a message to that mailbox. Messages are typically implemented as pointers to dynamically allocated objects or structures.

Events are used to synchronize the activities of groups of processes. An event is a variable with two states: occurred and not-occurred. If a process *waits* for an event which is in the occurred state, the process continues and the event is automatically returned to the not-occurred state. If a process waits for an event which is in the not-occurred state, the process will "wait" until that event is *set* (put in the occurred state) by another process. An alternative to the *wait* operation is the *queue* operation. A *set* operation for an event reactivates one process "queued" at the event and all of the processes "waiting" at the event.

CSIM *processes* are C (or C++) procedures which execute a *create* statement. CSIM18 provides its own thread (lightweight process) environment. Processes can be either *active* (computing), ready to become active, *holding* (allowing simulated time to pass), or in a queue (waiting for something to happen). This process environment is very straightforward and many users have found it to be easy to master and use to implement their models.

3 DATA COLLECTION

Data is automatically collected at all of the facilities and storages in a model. For each facility in a model, the data collected is used to report on the following output values:

- average service time
- server utilization
- process throughput rate
- average queue length (including processes in service)
- average response time (queueing delay plus service time), and
- the number of completed processes.

For a facility with multiple servers, there are data for each server and then summary data for the entire facility. A similar set of data is collected for each storage in a model.

There are data reporting routines for both facilities and storages. These can be used to generate a printed report for a specified facility or storage or for all of the facilities and/or storages in a model. The *report* procedure produces a printed report which summarizes the usage of all of the facilities and storages in a model.

Figure 1 is an example of a report for a system model with five facilities and a storage. The first facility is named “CPU” and has two servers. The remaining facilities are a collection of separate disk units. The storage is named “memory”.

Every reported data value can be accessed by using the "inspector" functions also provided with CSIM18. In addition, the data which is the input to the summary items is also accessible via these inspector functions. These inspector functions can be used to produce reports tailored to specific applications. Alternatively, these inspector functions can be used to create output files which can be input to other applications, including spreadsheets and other data analysis applications.

In addition to the data automatically collected for the facilities and storages, CSIM18 provides objects which

```
                CSIM Simulation Report (Version 18 for MSVC++ V4.0)

                                Tue Jul 15 19:13:58 1997


                              Ending simulation time:     25071.009
                              Elapsed simulation time:    25071.009
                              CPU time used (seconds):       58.490

FACILITY SUMMARY

facility     service    service          through-    queue     response   compl
name           disc      time    util.     put       length      time     count
-------------------------------------------------------------------------------
cpu          pre_res    0.06753  1.415   20.95352    1.85849    0.08870   525326
   > server  0          0.06791  0.778   11.45347                         287150
   > server  1          0.06706  0.637    9.50006                         238176
disk[0]      fcfs       0.02992  0.108    3.60344    0.11759    0.03263    90342
disk[1]      fcfs       0.02994  0.108    3.60628    0.11782    0.03267    90413
disk[2]      fcfs       0.03010  0.109    3.61461    0.11874    0.03285    90622
disk[3]      fcfs       0.02999  0.108    3.61741    0.11853    0.03277    90692

STORAGE SUMMARY

storage                 alloc            service     queue     response   allocs
name            size    amount   util.    time       length      time     compl
-------------------------------------------------------------------------------
memory           100     27.690  0.541    2.55581    3.60785    4.52263    20000
```

Figure 1: Facilities and Storage Report

are used to collected data specified by the model builder. There are four of these objects, as follows:

- tables,
- qtables,
- meters, and
- boxes

In more detail, a *table* is used to collect real-valued data items. A table is usually instantiated at the beginning of a run of a model. During the run, individual data items are *recorded* in the table. Then at the end of the run, a *report* is generated from the data summarized in the table. The summary for a table consists of several items including the following:

- mean,
- variance (and standard deviation),
- minimum,
- maximum, and
- the number of observations.

There are inspector functions to access the contents of tables including the summary items.

The data collected in a table can be augmented in three ways; for a table, any or all of the following enhancements can be specified:

- a *histogram* can be specified, to report on the relative frequency of pre-specified ranges of values
- a *moving window* can be specified, to use only the most recent "n" data values for reporting purposes (as opposed to all of the collected data values for the standard report), and
- *confidence intervals* can be specified; a confidence interval is used to estimate the accuracy of the estimate of the mean value collected in a table.

Confidence intervals are reported for three commonly used confidence levels: 90%, 95% aand 98%. The calculation of these confidence intervals uses the method of batch means. A heuristic, based on the computed autocorrelation, is used to calculate the length and number of the batches.

A *qtable* is used to collect data about time-based, integer-valued variables such the queue length at a server. The data is entered in the qtable by noting when, in the simulated operation of the system, the variable takes on new values. One way of entering this data is to note entries to and exits from the queue (or collection of states). This kind of variable is often called a state variable in a queueing system. The report for a qtable includes the following items:

- initial value of the state variable,
- final value,
- number of entries to the queue,
- number of exits,
- minimum value,
- maximum value,
- average value, and
- the variance and standard deviation.

As with tables, there are inspector functions for all of the items associated with qtables.

The data collected in a qtable can be augmented by adding the following enhancements:

- a histogram (to display the frequency of the different states), and
- a confidence interval.

A confidence interval helps assess the statistical accuracy of the mean value of the state variable. These are computed using the technique described for confidence intervals for tables.

A *meter* is used to "measure" the flow rate of processes past a specified point in the model. An example of this could be the rate at which processes terminate. A meter collects data on the flow rate. In addition, a meter includes a table which records interpassge times at the metering point.

A *box* captures data on the entry to and exit from a specified portion of the model. A box consists of a table (to collect data on process residency times "in the box") and a qtable (to collect data on the number of processes "in the box").

Figure 2 is a diagram showing the system model used to collect the data in Figure 1. This is model of a computer system processing a stream of jobs. Arriving jobs allocate a block of memory and then enter a cycle of computing (both at the CPU and a disk unit). When the job completes one or more cycles, it deallocates the block of memory and departs as a completed job.

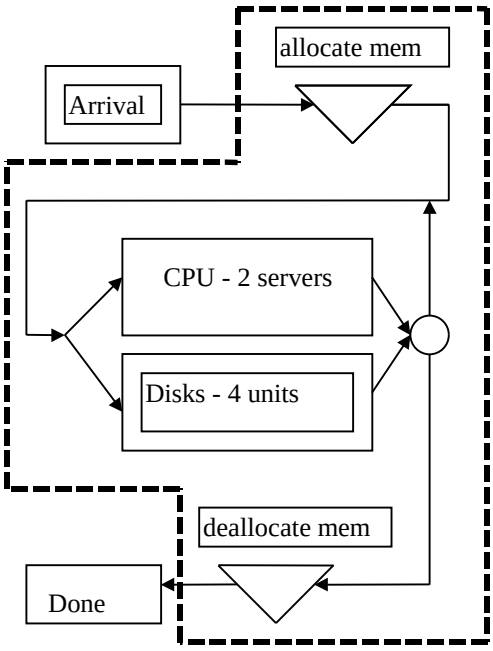

Figure 2: System with Box Added

The dark dashed line in Figure 2 marks the "box" used to collect data on the residency time and number of jobs in the active part of the model. The program has two statements which correspond to "entering" the box and

```
BOX 1:  system

      statistics on elapsed times

      minimum         0.043622          mean                   4.522627
      maximum       171.239097          variance              88.227782
      range         171.195475          standard deviation     9.392964
      observations       20000          coefficient of var     2.076882

                                                cumulative
      lower limit     frequency    proportion   proportion

          0.00000         10993     0.549650     0.549650   ********************
          2.00000          4111     0.205550     0.755200   *******
          4.00000          1478     0.073900     0.829100   ***
          6.00000           833     0.041650     0.870750   **
          8.00000           569     0.028450     0.899200   *
         10.00000           391     0.019550     0.918750   *
         12.00000           300     0.015000     0.933750   *
         14.00000           191     0.009550     0.943300   .
         16.00000           148     0.007400     0.950700   .
         18.00000           123     0.006150     0.956850   .
         20.00000           111     0.005550     0.962400   .
         22.00000            86     0.004300     0.966700   .
         24.00000            82     0.004100     0.970800   .
         26.00000            55     0.002750     0.973550   .
         28.00000            68     0.003400     0.976950   .
   >=    30.00000           461     0.023050     1.000000   *

      statistics on population

      initial       0      minimum       0      mean                   3.607854
      final         0      maximum      29      variance              15.200360
      entries   20000      range        29      standard deviation     3.898764
      exits     20000                           coeff of variation     1.080632

                                          cumulative
        number    total time    proportion   proportion

             0    8878.93561     0.354152     0.354152   ********************
             2    7337.57264     0.292672     0.646823   ***************
             4    3539.94939     0.141197     0.788020   ********
             6    1966.31190     0.078430     0.866450   ****
             8    1231.96129     0.049139     0.915589   ***
            10     814.25687     0.032478     0.948067   **
            12     483.36262     0.019280     0.967346   *
            14     285.67951     0.011395     0.978741   *
            16     215.02911     0.008577     0.987318   .
            18     138.07611     0.005507     0.992825   .
            20      92.30840     0.003682     0.996507   .
            22      46.06247     0.001837     0.998345   .
            24      26.53242     0.001058     0.999403   .
            26      13.00927     0.000519     0.999922   .
            28       1.96112     0.000078     1.000000   .
```

Figure 3: Box Report with Histograms

"exiting" the box. The data reported for this box is shown in Figure 3. This report gives the residency time in the box (the time in the system) and also summarizes the number of jobs in the box (queue length). The report gives both statistical summaries and frequency histograms for these data.

4 CONFIDENCE INTERVALS

In addition to statistical summaries with histograms, the CSIM18 data collection facility can also produce confidence intervals for a summarized data item. The method of batch means [[Law and Kelton, 1982] is used to compute confidence intervals. The standard report gives confidence levels for confidence levels of 90%, 95% and 98% respectively. Figure 4 is a confidence interval report for the example in Figure 2. For this report, the model simulated 20,000 jobs being processed (as opposed to 2,000 jobs reported in Figure 3).

model converged to the specified levels of accuracy or exceeded the CPU time limit.

Figure 5 is the output produced by invoking the run-length control feature in CSIM18. The feature uses a built-in event, named "converged", to allow the control program to wait until the specified accuracy has been achieved or until a cpu time limit has been exceeded. The call which invokes this run-length control is as follows:

```
systemBox.time_run_length(0.05, 0.90, 50.0);
```

This call specifies an accuracy of 0.05, a confidence level of 0.90 and a cpu time limit of 50.0 seconds.

In order to make use of ths feature, the model must be modified so that it generates an unlimited number of jobs. Normally, the model either executes until a pre-specified number of jobs have been processed or until a pre-specified amount of simulated time has elapsed.

```
confidence intervals for the mean after 20000 observations

level                confidence interval                rel. error

 90 %    4.522627 +/- 0.294710 = [4.227917, 4.817337]    0.069706
 95 %    4.522627 +/- 0.351673 = [4.170953, 4.874300]    0.084315
 98 %    4.522627 +/- 0.418229 = [4.104397, 4.940856]    0.101898
```

Figure 4: Confidence Interval Report

5 AUTOMATIC RUN-LENGTH CONTROL

Many simulation models are used to estimate one or more output parameters. These are termed *steady state* simulation models [Law and Kelton, 1982]]. Determining the accuracy of an estimate is typically accomplished by using confidence intervals.

CSIM18 allows the model builder to structure the model so that it will continue to execute until a specified accuracy and confidence level for an output parameter is achieved. Because some models may require a very long time to achieve the specified levels, a maximum CPU execution time is also specified. When the model terminates, the report informs the user whether or not the

6 SUMMARY

This paper has presented the data collection and run-length control features available in the CSIM18 simulation engine. These features simplify the collection of statistical data. Data describing the usage of facilities are automatically collecting and reported as is data describing usage of storages. In addition, tables, qtables, meters and boxes can be used to collect data tailored to the specific model. Finally, the automatic run-length control feature can be used to assure that statistically valid data is being presented, and that this data is collected in an economical manner.

```
results of run length control using confidence intervals

cpu time limit      50.0          accuracy requested      0.050000
cpu time used       14.0          accuracy achieved       0.046874

> the requested accuracy has been achieved

90.0% confidence interval: 2.333472 +/- 0.104482 = [2.228990, 2.437955]
```

Figure 5: Run-length Control Report

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.

REFERENCES

Law, A. and D. Kelton. 1982. *Simulation Modeling and Analysis*. 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.

AUTHOR BIOGRAPHIES

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.

JEFF BRUMFIELD is a Senior Leturer in the Department of Computer Sciences at The University of Texas at Austin. He received his Ph.D. in Computer Sciences from Purdue University in 1982. He has presented many professional development seminars and courses, covering the areas of system performance analysis, system modeling and simulation and distributed computing.