

Paper 317-2009

Exploring System Performance with SAS® Simulation Studio

Ed Hughes, Hong Chen, Emily Lada, Phil Meanor
SAS Institute Inc., Cary, NC

ABSTRACT

Discrete event simulation is used to model, study, plan, and improve systems in which random events play a dominant role. These systems are often driven by complicated mathematical and logical relationships, making it impossible to derive an analytical solution. The journal *OR/MS Today* cites simulation software as one of the most widely used tools in operations research, due in part to its applicability in a wide variety of fields including manufacturing, customer service, and health care.

SAS Simulation Studio (currently experimental in SAS/OR 9.2, with a production release planned for the third quarter of 2009) is a new Java-based application for modeling and analyzing systems through the use of discrete event simulation. Its graphical user interface requires no programming and provides a full set of tools for building, executing, and analyzing the results of discrete event simulation models. SAS Simulation Studio is designed to integrate with both SAS and JMP® for statistical analysis of simulation results; it also can interface with JMP to generate experimental designs.

SAS Simulation Studio is a flexible discrete event simulation application that provides extensive modeling and analysis tools suitable for both novice and advanced simulation users.

INTRODUCTION

This paper discusses the major features and functions of SAS Simulation Studio. We begin by providing some background on the nature and benefits of discrete event simulation as an analytical modeling method, highlighting its links to other forms of analysis. Turning to SAS Simulation Studio, we explain the motivation behind its creation and explore its current feature set, including the elements of the graphical user interface. We also consider the interactions between SAS Simulation Studio and JMP and the mutual benefits that result. In the appendix we walk through creation, execution, and statistics collection for a sample discrete event simulation model with SAS Simulation Studio.

WHAT IS DISCRETE EVENT SIMULATION?**MODELING THE REAL WORLD**

Simulation can be a wide-ranging term, applied in many industries and referring to many different forms of analysis. In perhaps its broadest sense, "simulation" denotes nothing more specific than the process of building a physical or logical model that mimics the behavior of a real-world system of interest. Physical models include wind tunnels and earthquake simulators, and a flight simulator combines physical and logical modeling. Here, however, we focus on purely logical models.

Often, by using statistical analysis and related methods, it is possible to uncover logical and mathematical relationships between the key elements of a system. Furthermore, sometimes it is possible to derive an analytical solution to the problem of how to make choices that result in the best possible system performance. In many cases, however, a great number of simplifying assumptions must be made in order to create a usable analytical model and obtain a precise solution. The end result of these simplifying assumptions might be the creation of a model that is too simple to be a realistic depiction of the original system.

Moreover, many real-world systems include not only complex mathematical and logical relationships but also significant random components. For such systems an analytical model, even a simple one, might not be possible. A far better approach is to incorporate the random elements of the system in the model. Discrete event simulation is one such modeling technique.

A prime motivation for building and running a discrete event simulation model is the creation of realistic data on the performance of the system being modeled, with an overall goal of using the data to make statistically valid inferences

about the performance of the system. Discrete event simulation offers the chance to generate such data without the need to create and observe the system in the real world. This is an attractive alternative in many cases. For example, building several possible versions of the real-world system might be impossible, impractical, or cost-prohibitive. In extreme cases (as in major construction), it might be true that no real version of the system exists. Some versions of the system might provide such poor service (as in medical settings) as to create unacceptable legal liabilities. In these and many other cases, discrete event simulation is clearly preferable to direct observation.

BASICS OF DISCRETE EVENT SIMULATION

In systems suited to analysis by discrete event simulation, the state of the system being modeled changes only when discrete events occur; the term “discrete events” signifies that each such event occurs at a specific time and lasts for a specific duration. Although the start time and duration for an event can both be random, the event does not occur continuously or go on indefinitely; likewise, the state of the overall system is not constantly changing. Instead, changes in the state of the system—and in the state of the model—occur only at a countable set of distinct points in time. In a model of an emergency room, the state of the model would change, for example, when a patient arrives, when a patient begins or completes treatment, or when a nurse or doctor begins or ends a meal break.

In a discrete event simulation model of the system of interest, a computer generates numerical data that imitates—or simulates—the random elements of the system. This includes, but is not limited to, time-related elements. In the emergency room example, simulated data might include not only arrival times for patients and durations of interactions with specific emergency room staff but also the type and severity of malady or injury for each arriving patient. As the simulation model runs, each event occurs at its randomly generated time and lasts for its generated duration; the computer keeps track of the occurrence of these events and also updates the state of the system whenever an event occurs.

Thus discrete event simulation offers at least two more distinct advantages over real-world observation, in addition to those cited in the previous section. First, data can be gathered at a very rapid rate since in a model it is easy to move through hours, days, weeks, or months of system performance in mere seconds; only the points at which the state of the system changes are of interest, and so all of the intervening time periods can be skipped. This speed of data generation contributes to the second additional benefit: you can quickly and easily create and gather data on many alternate scenarios, each representing a combination of specific operating conditions, assumptions, policies, configuration choices, and so on.

DISCRETE EVENT SIMULATION MODELING

Often discrete event simulation is used to model queuing systems, in which entities (representing objects or individuals) are receiving some sort of time-consuming service, are waiting to receive service, or are otherwise moving through the system. The term “queuing” comes from the word “queue,” a synonym for a “line” in which one waits. Examples of these systems include the aforementioned emergency room (“service” might represent identification of injury or malady, verification of insurance coverage, or various forms of medical treatment or consultation), a retail checkout area (“service” is the time spent with a cashier), or a bank lobby (“service” is provided by a bank teller or officer of the bank).

The overall structure of a discrete event simulation model resembles a flowchart or network; it is represented by nodes linked by directed arcs which indicate the flow of entities through the system being modeled. Usually these nodes represent the entities’ experiences in the system: entering or exiting, waiting in a queue, spending time while receiving service, choosing or accepting an assigned choice among several alternatives, and so on. Movement along arcs that link the nodes can represent physical movement between locations or can represent a transition from one status to another—for example, from “waiting for service” to “receiving service.” In a model of an emergency room, for example, movement along an arc might represent movement from a treatment area to the post-treatment payment area or might instead represent the start of a consultation, with no corresponding change in physical location.

THE ROLE OF EXPERIMENTAL DESIGN

The flowchart or network structure of a discrete event simulation model makes it much easier to ensure that the key relationships are represented accurately, but for effective use of the model more is needed. Realistically, once you are confident that your simulation model accurately reflects the system under investigation, you need an effective and efficient method for studying the effects of changes in selected control settings on the performance of your model. The structured investigation method provided by experimental design (or, equivalently, design of experiments) helps to accomplish this.

Usually you have several key performance metrics in mind. Examples might include the average wait for service, the average length of a queue, or the number of customers served. System controls might include staffing levels, rates of work, and maximum lengths for queues. The performance metrics are referred to as “responses” and the controls as “factors.”

Once you have defined factors and responses, you need to create multiple versions of your simulation model to account for the possible combinations of factor values. Each version of the model might need to be run several times in order to generate enough data to support valid statistical inferences about the factors’ influence on the responses. Each unique combination of factor values, coupled with the desired number of runs, is termed an “experiment” or “design point.” The principles of experimental design guide you in creating a set of experiments to generate system performance data that supports statistical inference.

WHAT IS SAS SIMULATION STUDIO?

SAS Simulation Studio is a Java-based SAS application in SAS/OR designed for modeling and analyzing systems through the use of discrete event simulation. SAS Simulation Studio, experimental in SAS/OR 9.2, is the successor to QSIM, SAS/OR’s longstanding discrete event simulation application. SAS Simulation Studio features a graphical user interface that requires no programming and provides all the tools needed for building, executing, and analyzing discrete event simulation models.

While a comprehensive set of modeling tools is an important attribute of a simulation software application, advanced analysis tools are arguably just as important. As mentioned above, analyzing output from discrete event simulations often can require the use of advanced statistical methods. SAS Simulation Studio is designed to interact with both SAS and JMP for statistical analysis of simulation results. Data generated by a model can be saved as a SAS data set or as a JMP table for later analysis.

GRAPHICAL USER INTERFACE ELEMENTS

The graphical user interface for SAS Simulation Studio, shown in Figure 1, uses a hierarchical structure to assist in organizing your work. At the top level of this hierarchy is the notion of a “project.” A project is a collection of models and experiments that correspond to each system being studied. Within each Project window you can create one or more Model windows in which you can build simulation models.

Another major element of the interface is the Block Template Display area, populated with modular blocks that correspond to the elements of a discrete event simulation model. The basic set of blocks provided with the application is extensive; it includes creation and disposition of entities, various types of queues, servers, routing and connections, data input and generation, numeric and graphical monitoring and reporting, and several other functional areas. You can create customized “compound blocks” by selecting and connecting any of these basic blocks, and you can store these new blocks for further use in modeling. This is especially useful if you need to replicate the same or very similar functionality at several points in a model.

For each Model window you can create one or more Experiment windows; these windows provide you with an organized way to characterize, plan, and coordinate multiple runs of your simulation models. At a minimum you can specify the start time, end time, and number of replications to run for each version of a model. If you have defined factors and responses for your model, you can use the Experiment window to investigate the effects of varying factor values on responses.

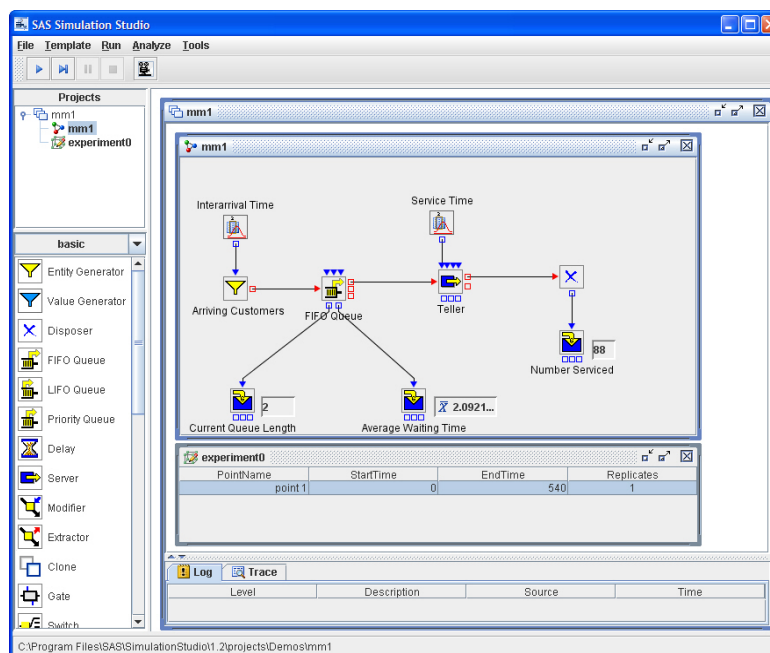


Figure 1. The SAS Simulation Studio Graphical User Interface

Construction of models with SAS Simulation Studio is accomplished by dragging block icons from the template into a Model window and connecting them with arcs created by drag-and-drop. Note that in this modeling environment there are two types of input and output ports attached to the blocks. Red ports, on the right and left sides of blocks, are used to pass entities between blocks via arcs. Blue ports, on the top and bottom sides of the blocks, are used to pass information between blocks, again via arcs. Examples of information flows include the number of active servers or the service time for a Server block, the time between entity arrivals for an Entity Generator block, and performance metrics (length of queue, wait time) from a Queue block.

Once you have built a model, you can run it by using the controls on the toolbar at the top of the SAS Simulation Studio window (Figure 1). In order to run a model you need an Experiment window with at least one design point in it. By default, for each model SAS Simulation Studio creates one design point in an Experiment window, listing the start time, end time, and number of replicates. Edit these fields if you want and then click the **Start** button (leftmost button, with the "Play" icon). For debugging purposes you might want to click the **Animate** button (rightmost button, with the movie camera icon) first to show the flow of entities and information through the model. Pause the run of the model at any time by clicking the **Pause** button (third button from the left). Stop and reset the model run by clicking the **Reset** button (second button from the right, with the "Stop" icon). These controls are also available via the **Run** pull-down menu, along with controls that enable you to view the simulation clock and the count of replications run for the model.

If you want to save your model, select the **Save As** option on the **File** pull-down menu. You can open a saved model with the **File > Open** option; you can choose to open an entire project, a specific model in a project, or a specific experiment associated with a model.

For a more detailed description of the basic use of the SAS Simulation Studio user interface, see the appendix, "Building a Running a Simple Queuing Model" or consult the SAS Simulation Studio User's Guide (see the "Availability" section later in this paper for details on how to obtain this documentation).

DESIGN OF EXPERIMENTS AND JMP INTEGRATION

Recall from the introductory section that, for a model with defined factors that represent controls on the configuration of the model and responses that represent system performance metrics of interest, the goal is to create and collect enough data to characterize the effects of the controls on the responses. This means that you must run simulations and collect the created data for a sufficient number of versions of the model and for a sufficient number of replications of each version. The specific choice of factor settings and replications for each version (or design point) is referred to as design of experiments (or, equivalently, experimental design).

SAS Simulation Studio, through the Experiment window, supports both manual and automated design of experiments. Manual design of experiments simply means that you determine which design points to create and how many replications should be run for each point. This is useful if you need to make a direct comparison of a small number of different versions of the model or if you want to carry out a highly specialized experimental design. To accomplish this, simply right click in the Experiment window, choose **Add Design Point**, and edit the column entries in the row created for the added design point.

For automated design of experiments, SAS Simulation Studio integrates with JMP statistical discovery software from SAS. To create an experimental design with JMP, right click in the Experiment window and select **Make Design**. The custom designer in JMP creates a suitable design and automatically passes it back to the Experiment window. If you want, you can alter this design in the JMP interface, perhaps by adding interaction terms between factors or by adding replications for specific design points. After making any such design changes in JMP, click the **Commit** button in the upper left corner of the JMP Design of Experiments window to pass the new design back to the Experiment window in SAS Simulation Studio.

Figure 2 shows an Experiment window for a repair shop model. There are three factors, shown in the yellow columns, which denote staffing levels (number of servers) at the Quality Control, Repair, and Service locations. There are seven responses, highlighted in pink: number of units fixed and, for each of Quality Control, Repair, and Service, metrics on the average wait and the average utilization of the servers. In this design, five replications are specified for each design point. In Figure 2, design point number 4 has been expanded to show the results for each of its five replications. The remaining rows display response figures that represent the mean of the results from their five replications.

PointName	StartTime	EndTime	NumQC	NumRep...	NumServ...	Replicates	NumFixed	AvgUtilQC	AvgWaitQC	AvgUtilRep...	AvgWaitServ...	AvgUtilServ...	AvgWaitRep...
point 1	0	2,700	1	1	3	▶ 5	55.8	98.117	640.598	32.901	0.114	22.777	1.682
point 2	0	2,700	3	2	1	▶ 5	105.6	61.846	3.659	16.395	16.765	67.899	0
point 3	0	2,700	2	3	1	▶ 5	100.2	88.317	84.874	10.93	16.765	67.899	0
point 4	0	2,700	2	1	3	▼ 5	100.6	88.508	97.37	32.901	0.114	22.777	1.682
						1	110	97.535	84.371	35.83	0.023	22.105	1.389
						2	98	86.191	21.891	29.272	0.029	21.946	1.253
						3	94	82.413	47.868	31.828	0.079	19.992	1.557
						4	93	82.178	23.499	29.027	0.116	21.037	1.68
						5	108	94.224	309.22	38.549	0.326	28.806	2.529
point 5	0	2,700	2	1	1	▶ 5	100.2	88.317	84.641	32.79	16.765	67.899	0.234
point 6	0	2,700	3	1	2	▶ 5	105.8	61.918	8.7	32.901	0.837	34.135	1.382
point 7	0	2,700	2	2	2	▶ 5	100.4	88.472	97.977	16.473	0.837	34.135	0.021
point 8	0	2,700	2	2	3	▶ 5	100.6	88.515	98.456	16.468	0.114	22.777	0.095
point 9	0	2,700	1	1	1	▶ 5	55.8	98.117	621.012	32.79	16.765	67.899	0.234
point 10	0	2,700	3	3	3	▶ 5	105.8	61.941	9.329	10.978	0.114	22.777	0.001
point 11	0	2,700	1	3	2	▶ 5	55.8	98.117	641.409	10.982	0.837	34.135	0
point 12	0	2,700	1	2	1	▶ 5	55.8	98.117	621.193	16.395	16.765	67.899	0

Figure 2. Experiment Window Showing Multiple Design Points

For any design, you run the replications associated with design points by highlighting the desired rows in the Experiment window and clicking the **Start** button on the SAS Simulation Studio toolbar. If you choose to run the entire design, then you should highlight all of the rows. You will see each selected design point's row highlighted with red text in turn as they are run; in the highlighted row the response column values are refreshed as each replication is run. After the run has completed, you can click the arrow in the Replicates column to expand the design point and make each replication visible.

ANALYZING SIMULATION RESULTS

In SAS Simulation Studio, several blocks¹ are dedicated to producing graphical analysis of the simulation results, both as the simulation model runs and at the termination of a run. These displays can be useful when debugging or tuning a model, or when just a single run of a model is desired.

¹ The Histogram, BarChart, Scatterplot, and Boxplot blocks provide graphical displays of simulation data. The Number Holder and String Holder blocks display data non-graphically.

SAS Simulation Studio also provides a number of blocks² that can collect data during simulation runs so that the data can be stored for later and more extensive analysis. You can configure how data is stored by selecting **Configuration** in the **Tools** pull-down menu at the top of the SAS Simulation Studio window. The resulting SAS Simulation Configuration window enables you to choose whether data is to be stored by default as SAS data sets or as JMP tables. You can then use either SAS or JMP, respectively, to carry out analysis of the stored data.

CONCLUSION

Discrete event simulation is an analytical approach that assists in the study and evaluation of systems with significant random elements and where direct measurement is not advisable, practical, or effective. This approach is growing steadily in importance due to its usefulness in a wide range of industries.

SAS Simulation Studio is a new graphically oriented application in SAS/OR that provides an excellent environment for building, running, and analyzing discrete event simulation models. This environment is intuitive enough to serve the novice user but also includes a number of features that enable the more experienced user to build and work with more sophisticated, more highly detailed models. The hierarchical project/model/experiment structure of the SAS Simulation Studio interface is invaluable in keeping expert or novice simulation work organized.

Because analysis of simulation results is critically important, SAS Simulation Studio provides easy integration with the extensive statistical analysis methods available in SAS and JMP. Finally, state-of-the-art experimental design capabilities, delivered via integration with JMP, help to ensure that your discrete event simulation work is focused and productive.

AVAILABILITY

SAS Simulation Studio is available in experimental form with SAS/OR 9.2. Updates will be made available via download at support.sas.com. Select **Software Downloads** from <http://support.sas.com/techsup/download/>; next select **SAS/OR Software** and finally select **SAS Simulation Studio**. The SAS Simulation Studio User's Guide also can be downloaded from this site. Note that SAS Simulation Studio requires Java 1.5 or newer.

APPENDIX: BUILDING AND RUNNING A SIMPLE QUEUING MODEL

To illustrate some of the basic concepts involved in building models in SAS Simulation Studio, we construct a model of a simple banking system with one teller. In this model, entities correspond to individual bank customers. We assume that customers arrive at the bank at a rate of 10 per hour (so that the interarrival time between customers is a sample from the exponential distribution with a mean of 6 minutes). Customers wait in a single line on a first-come, first-served basis (also termed "first in, first out" and abbreviated "FIFO"). We also assume that the teller has a service rate of 12 customers per hour (so that the service time for each customer is a sample from the exponential distribution with a mean of 5 minutes). This simple banking system is an example of what is referred to as an M/M/1 queuing system (exponential interarrival time, exponential service time, one server).

For a queuing system such as this one, the following statistics might be of interest:

- the average time a customer spends waiting in line
- the length of the queue
- the number of customers served in one day

Figure 3 shows a SAS Simulation Studio model of the banking system. All the blocks used in this example can be found on the basic template of blocks provided with the application.

² The Bucket, Number Holder, Probe, Queue Stats Collector, and Server Stats Collector blocks in SAS Simulation Studio can collect data.

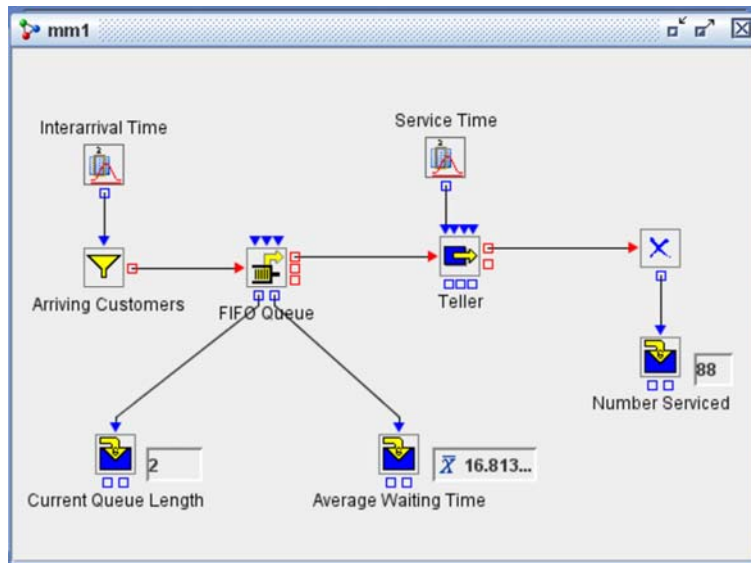



Figure 3. An M/M/1 Queuing Model of a Bank Lobby

Customer arrivals to the bank are modeled using the Entity Generator block  labeled “Arriving Customers.” The Entity Generator block has one input *value port* for the interarrival time of entities. This is the InterArrivalTime port.




The Numeric Source block  labeled “Interarrival Time” generates a sample from the exponential distribution (representing the next interarrival time), and the Entity Generator pulls that value through the InterArrivalTime port. Figure 4 shows the dialog box for the Numeric Source block, indicating that interarrival times are sampled from an exponential distribution with mean 6.



Figure 4. Numeric Source Block: Dialog Box

The simulation clock, which tracks the progress of a simulation model run, has no defined units. You can designate units of simulation time either as hours or as minutes (or, in truth, as any time unit), as long as you use the same units consistently throughout the model. We choose to designate simulation clock units as minutes.

QUEUING

When an entity that represents a customer leaves the Entity Generator block, it is pushed down an arc connected to the FIFO Queue block . It is important to note that the movement of an entity down this arc does not advance the simulation clock because it does not correspond to any time-consuming physical movement. If the FIFO Queue has a limited capacity and it is full when an entity arrives, the entity is pushed out the queue OutBalk port. If the FIFO Queue is not full, the FIFO Queue block attempts to push the entity to the Server block  labeled "Teller" which represents the bank teller. If the Server is available, it accepts the entity. Otherwise, the entity waits in the FIFO Queue. When the Server becomes available (that is, when the teller is not assisting a customer), it requests an entity from the FIFO Queue.



SERVICE


When the entity arrives at the Server block , a service time is sampled from the second Numeric Source block labeled "Service Time" and is pulled by the Server through the InServiceTime port (to determine how much time the teller will spend with this customer). Once the entity completes its service, it is pushed out to the Disposer block  where it leaves the system. The Server then requests another entity from the FIFO Queue.

RUNNING THE MODEL

Figure 5 shows the Experiment window for this model. There is one experimental design point called *point1* for which the number of replications is set to 1 and the length of the simulation is set to 540 minutes (one banking day).

To visualize the simulation clock, select **Show>Simulation Clock** from the **Run** pull-down menu. To turn on the animation, click the **Animation** button .

To run this model, click the **Start** button . The model can be paused by clicking the **Pause** button . Clicking the **Start** button restarts the model.

When the model has completed its run, only the **Reset** button  is active. The **Reset** button must be pushed before making changes to the experiment window or rerunning the model.

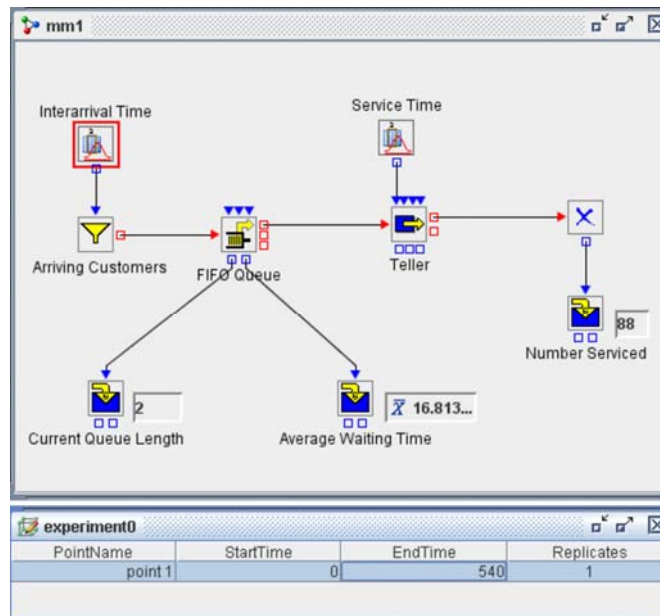


Figure 5. M/M/1 Queuing Model: One Design Point

STATISTICS COLLECTION


Number Holder blocks  can be used to collect and display statistics such as minimum, maximum, sum, and mean as the model is running. In Figure 5, a Number Holder block labeled "Average Waiting Time" is connected to the OutWait port on the FIFO Queue.

Figure 6 shows the dialog box for this Number Holder block. As each entity leaves the FIFO Queue, its wait time is pushed into the Number Holder block, whose Display field is set to Mean. The Number Holder then recomputes the average waiting time and displays the new value. In this example, the average waiting time for customers computed over one banking day is 16.813 minutes (as seen in the “Average Waiting Time” Number Holder display in Figure 5).

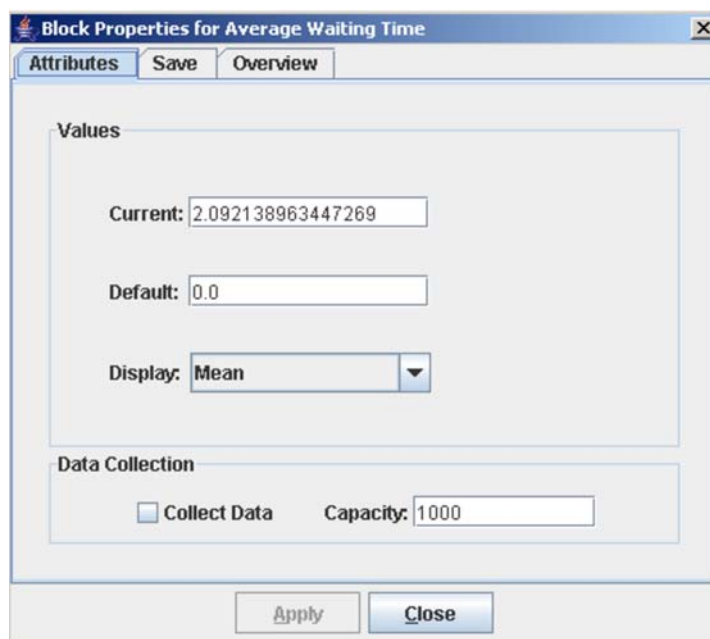


Figure 6. Number Holder Block Dialog Box

A second Number Holder block, labeled “Current Queue Length,” with the Display field set to Value, is connected to the OutLength port on the FIFO Queue. Each time an entity enters or leaves the queue, the new queue length is pushed to the Number Holder block and the updated queue length is displayed.

Note that Number Holder blocks can display only averages for observation-based statistics, such as waiting time. For time-dependent statistics like queue length, the Number Holder should be used only to display the minimum, maximum, sum, or current value. Figure 5 shows that the final queue length is 2.

Finally, there is a third Number Holder block, labeled “Number Served,” with the Display field set to Value, connected to the OutCount port on the Disposer block. Each time an entity leaves the system, the Number Holder updates its value and displays the current number of entities serviced. Figure 5 shows that in this example the number of customers served by the end of one day is 88.

REFERENCES

SAS Institute Inc. 2007. *SAS Simulation Studio User's Guide*. Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Ed Hughes
 SAS Institute Inc.
 SAS Campus Drive
 Cary, NC 27513
 Work Phone: (919) 531-6916
 E-mail: Ed.Hughes@sas.com

Hong Chen
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Work Phone: (919) 531-6091
E-mail: Hong.Chen@sas.com

Emily Lada
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Work Phone: (919) 531-1391
E-mail: Emily.Lada@sas.com

Phil Meanor
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Work Phone: (919) 531-6043
E-mail: Phillip.Meanor@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.