Paper 238-2010

Modeling Systems and Resources with SAS® Simulation Studio: An Introduction

Ed Hughes
SAS Institute Inc., Cary, NC

ABSTRACT

Discrete event simulation is used to model and investigate systems with complicated mathematical and logical relationships that make analytical solutions impossible—often featuring significant random elements and events. Simulation, one of the most widely used methods in operations research, captures these random elements to enable you to study system performance under varying conditions and configurations. It is employed in a wide variety of fields including manufacturing, customer service, and health care.

SAS Simulation Studio is a graphical discrete event simulation application, available as part of SAS/OR® software and also as an add-on module for JMP® software (as SAS® Simulation Studio for JMP).

Resources form a critical element of almost all simulation models. These might be unmovable resources such as large manufacturing machinery, classrooms, or treatment rooms in a hospital—or they might be movable resources such as physicians and nurses, portable equipment, or vehicles. SAS Simulation Studio provides unparalleled support for modeling with both stationary and mobile resources, enabling you to describe with great realism and detail the essential role that resources play in system performance.

INTRODUCTION

This paper surveys the resource modeling capabilities of SAS Simulation Studio. After some brief background information about the nature and benefits of discrete event simulation and a short overview of SAS Simulation Studio, we discuss the central role that resources play in simulation modeling. We describe and compare the different types of resources that you can model with SAS Simulation Studio, with an emphasis on SAS Simulation Studio's unique ability to model mobile resources.

Next we focus on the creation, use, and tracking of resources in SAS Simulation Studio models. We cover the elements of the basic "life cycle" of resources, from creating resources to disposing of them, and we also touch on more sophisticated optional functionality including scheduling resource availability and collecting statistics on resources while a model is running.

Prior exposure to SAS Simulation Studio specifically and to discrete event simulation in general is helpful but not critical in reading and benefiting from this paper. Readers who fulfill these requirements might wish to skip either or both of the next two sections and begin reading at the section "Resources in Simulation Models."

DISCRETE EVENT SIMULATION MODELING

Systems across a wide range of industries can be modeled using discrete event simulation, but these seemingly disparate systems share at least one important characteristic—the state of the system changes only when one or more discrete (separable) events occur. The term "discrete event" refers to the fact that an event occurs at a specific time and lasts for a specific duration—even if both are random. Just as the event does not occur continuously, the state of the system doesn't change continuously. State changes occur only when events occur. If you are simulating an emergency room in a hospital, events would include patient arrival, a patient starting or ending treatment, patient departures, and medical staff starting or ending a break period.

In a discrete event simulation model, 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.

Discrete event simulation often models queuing systems, in which "entities" (representing objects or individuals) move through the system and generally are either receiving some sort of time-consuming service or are waiting to receive service. Examples include an emergency room (in which various clerical or medical services are provided to patients), a retail store's cashier area ("service" is time spent with the cashier), or an insurance company's claims processing operations (claims are serviced by being evaluated at various stages). Discrete event simulation enables

you to investigate the system's performance for many different scenarios, defined by system configuration choices or operating conditions or both.

Graphically the overall structure of a discrete event simulation model can resemble a flowchart, with nodes linked by directed arcs that represent the flow of entities through the system. The nodes correspond to the entities' experiences in the system: waiting, receiving service, being directed to a location based on preferences, and so on. Movement along the arcs can represent physical movement in the system or a change in status—for example, from waiting for service to receiving service.

Entities, as they move through a simulation model, frequently need to carry along with them pieces of descriptive information such as a part number, a customer preference, a classification for a package, and so on. These pieces of information, known as attributes, are used at various points in the simulation model to control the routing and handling of the entities.

SAS SIMULATION STUDIO

SAS Simulation Studio is a Java application that provides a graphical interface for building, running, and analyzing discrete event simulation models. No programming is needed, and the interface is suitable for both novice and advanced simulation modelers. With SAS Simulation Studio you build models by using drag-and-drop commands to choose blocks (representing model components) from one of several templates and create instances of the chosen blocks in a modeling window. You connect the appropriate ports on your selected blocks with arcs that indicate the flow of entities and information between the blocks. Pop-up dialog boxes associated with the various blocks enable you to customize the block's functionality. SAS Simulation Studio is provided as a component of SAS/OR and can be licensed as an add-on product to JMP software (as SAS Simulation Studio for JMP).

While a comprehensive set of modeling tools is important in simulation software, advanced analysis capabilities are critical as well. Since analyzing the data generated by discrete event simulation models often requires the use of advanced statistical methods, SAS Simulation Studio is designed to interact with both SAS® and JMP software for statistical analysis of simulation results. Data generated by a simulation model can easily be saved as a SAS data set or a JMP table for later analysis, and SAS Simulation Studio also provides the means for integrating statistical analyses into the running or post-run processing of a simulation model.

The graphical user interface for SAS Simulation Studio, shown in Figure 1, has many features that assist you in the building, running, and analysis of simulation models.

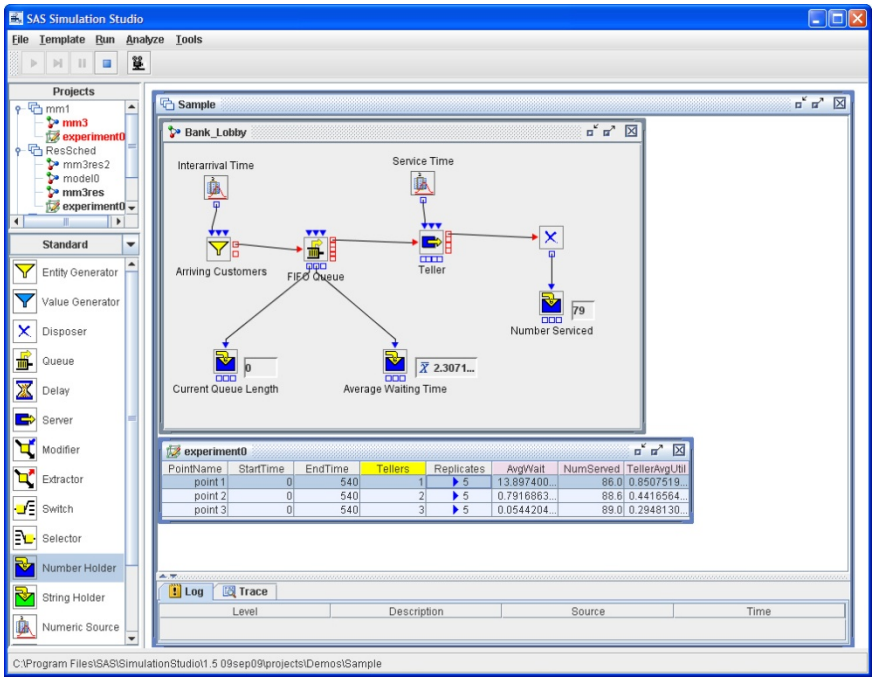

Figure 1. The SAS Simulation Studio Graphical User Interface

See the SAS Global Forum 2009 paper 317-2009, “Exploring System Performance with SAS Simulation Studio,” for more details on all of the aforementioned SAS Simulation Studio features.

RESOURCES IN SIMULATION MODELS

In simulation models and in real systems, resources provide support or aid on an as-needed basis. This support can take many forms including services, supplies, and facilities. Examples of resources include a worker that performs an assembly operation, a stockpile of parts used in the assembly operation, a treatment room in a hospital emergency facility, a vehicle that transports materials, and a teller who handles financial transactions for a bank client. Some resources can be modeled as having unlimited availability while others might be available only on a limited basis; the availability of a resource might easily vary during the run of a simulation model. Some resources might have a fixed location while others are able to move. In all cases, resources and their availability can significantly influence the flow of entities through a simulation model.

MODELING RESOURCES WITH SAS SIMULATION STUDIO

SAS Simulation Studio provides you with the ability to model both stationary and mobile resources. Stationary resources are modeled using blocks that can hold entities; these include the Queue, Server, and Delay blocks. The resources modeled with these blocks are created when the model is built, and they persist throughout the run of the model. Mobile resources, in contrast, are not modeled using blocks. Instead mobile resources are modeled using a special type of entity, the resource entity, which possesses all of the capabilities and attributes of regular entities. These resource entities are created and can be disposed of during the run of the simulation model, flowing through the model just as regular entities do. For the most part, resource entities can be processed and managed by the same blocks used for regular entities.

Like other entities in SAS Simulation Studio, resource entities can carry attributes—pieces of descriptive information that can be used to manage the routing and handling of an entity at various points in the model. Additionally, since resources often have a capacity (indicating number of units) associated with them, all resource entities carry a predefined attribute named ResourceUnits. Although there are special uses for the ResourceUnits attribute, it can also be used as an ordinary numeric attribute for modeling purposes. Resource entities also carry information that changes automatically during the run of the simulation model. This information includes seized status (whether the resource being used by a regular entity to provide service) and state (Functional or Nonfunctional; Nonfunctional includes Failed, Maintenance, and Offline).

COMPARING STATIONARY AND MOBILE RESOURCES

As a point of reference, let’s look first at a simple simulation model of a bank lobby, using a stationary resource. The model is shown in Figure 2.

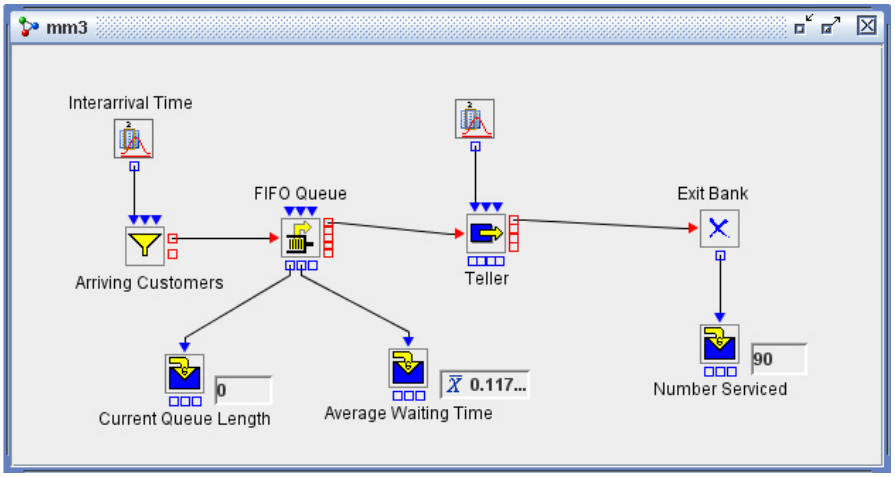

Figure 2. Bank Lobby Model, Using a Stationary (Server) Resource

In this model customers arrive, wait in a FIFO (first in, first out) queue for an available teller, conduct their business with a teller when one becomes available, and then leave the bank. The model tracks the current length of the queue, the average waiting time in the queue, and the overall number of customers served. An Entity Generator block creates the arriving customer entities, a Queue block (with FIFO discipline selected) handles the queuing, and a

Disposer block facilitates the exit of the customer entities from the system. The teller is modeled by a stationary resource, a Server block with a capacity of three (representing three tellers).

Now let's see how we could model the same bank lobby system with mobile resources. Figure 3 shows the revised model.

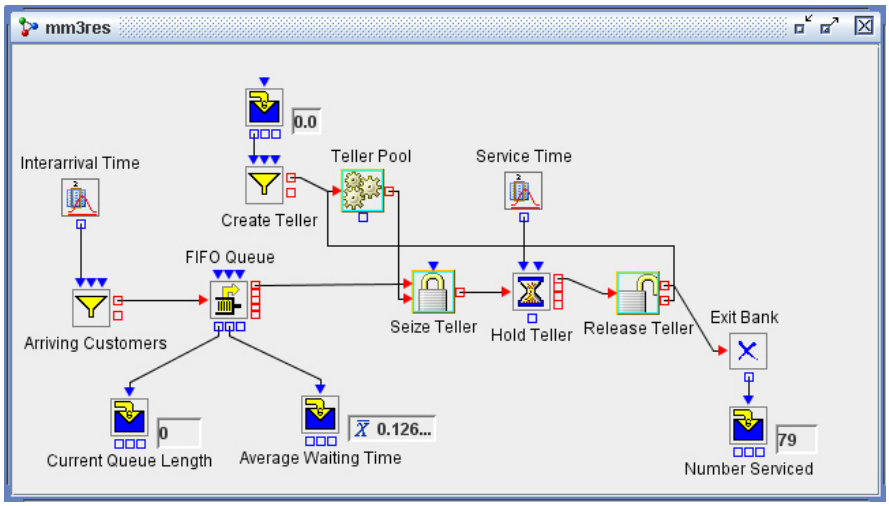

Figure 3. Bank Lobby Model, Using Mobile Resources

Arrival, queuing, and exiting in the two models are identical; the differences arise chiefly in the area between the Queue block and the Disposer block, where the tellers provide service to the customers. In this model we replicate all of the functionality of the previous model and also add an element that enhances the realism of the model.

In the new model the three tellers are modeled as resource entities named "BankTellers," which have been defined for this model using the Entity Types dialog box as shown in Figure 4.

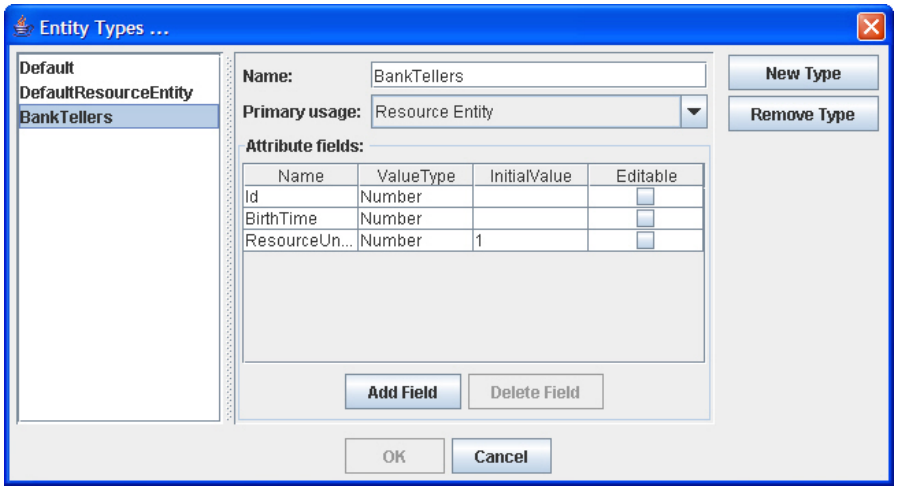

Figure 4. Entity Types Dialog Box

SAS Simulation Studio provides defaults for both regular entities and resource entities; you can copy and modify these defaults as needed to create your own entities. In the preceding example we've created the resource entity "BankTellers." Now we can use the Entity Generator block named "Create Tellers" to create the BankTeller resource entities in the revised model. Three BankTeller entities are created at time zero (the start of the run of the simulation model). Upon creation, the BankTeller entities move directly to the ResourcePool block labeled "Teller Pool" as

shown in Figure 3, waiting there until requested by a customer. The Seize block labeled “Seize Teller,” connected to both the Resource Pool and the Queue block, is used to pair a BankTeller entity with a customer entity as needed.

When a customer entity reaches the front of the FIFO queue, the Queue block notifies the Seize Teller block, which in turn checks to see whether one BankTeller entity is present in the Teller Pool block. If not, the customer entity continues to wait in the FIFO queue; otherwise one BankTeller entity is pulled from the Teller Pool and is attached to the customer entity in the Seize Teller block. Next, the paired entities move to a Delay block which holds them for a period of time, representing the time required for the customer to conduct his or her business with the teller. After this time has elapsed, the entities proceed to a Release block labeled “Release Teller.” Here the BankTeller entity is freed from the customer entity, and the customer entity moves next to the Disposer block labeled “Exit Bank.”

These two models illustrate, on a basic level, the essential distinction between modeling with stationary resources and modeling with mobile resources. However, you might wonder why we’ve taken a fairly simple model and have complicated it, replacing a single Server block with seven new blocks when the two models appear to do precisely the same thing. The answer is at least twofold. First, the use of resource entities instead of a stationary resource (the Server block) in modeling the tellers gives you finer control over the creation and use of resources in your models. Although this isn’t readily apparent in a simple model as the bank lobby example, the benefits become evident as you work with more complex models. These benefits include:

- the ability to seize multiple resources simultaneously
- the ability to preempt resources during service
- the ability to release partial resources at different stages
- the ability to collect statistics on selected resources
- the ability to route resources to different locations

The last point can be illustrated by considering another version of this model, as shown in Figure 5.

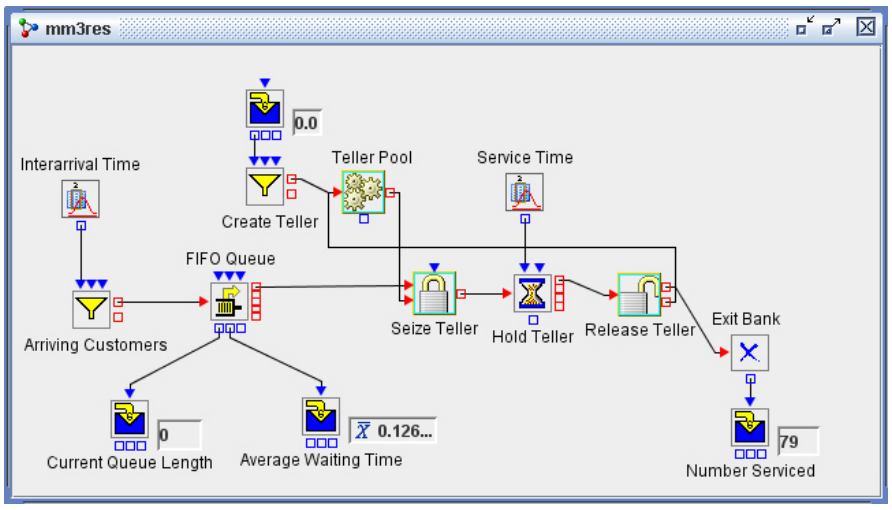

Figure 5. Bank Lobby Model with Additional Teller Task

Recall that in the Release Teller block the customer entity and BankTeller entity are separated. In the preceding model the BankTeller entity then proceeded directly to the Teller Pool block and instantly became available to be paired with another customer entity; in effect this is what happens also in the original model. In this model we choose to have the teller complete some clerical tasks, perhaps logging the details of the previous transactions, before preparing to serve another customer.

After exiting the Release Teller block, the newly released BankTeller entity proceeds to a Delay block labeled “Log Entry.” Here a period of time that represents the time needed to complete the log entry elapses, after which the BankTeller entity returns to the Teller Pool block. In more complex models there might be additional tasks for a mobile resource entity to perform before returning to a Resource Pool block. SAS Simulation Studio enables you to capture this behavior and to model its impact on resource availability, ultimately creating a more realistic model.

ADDITIONAL RESOURCE FEATURES

In order to explore some of the more advanced functionality that is available to models using resource entities in SAS Simulation Studio we continue with the bank lobby example, adding some useful capabilities. The revised model is shown in Figure 6.

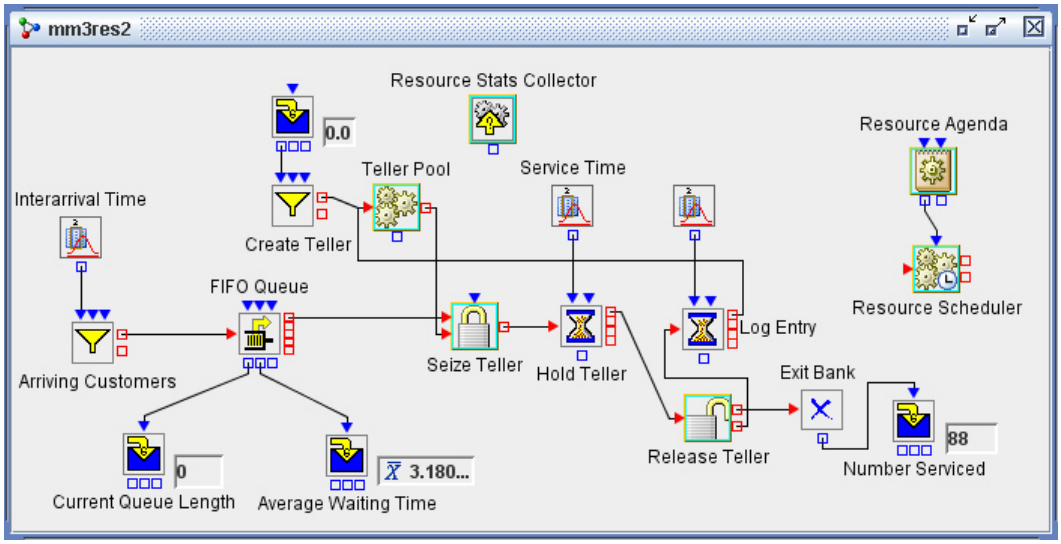

Figure 6. Bank Lobby Model, Adding Resource Scheduling and Resource Statistics Collection

The three new elements in this model are the Resource Agenda and Resource Scheduler blocks at the upper right and the Resource Stats Collector block at the top center.

SCHEDULING RESOURCE ENTITIES

In previous versions of this model the tellers, represented by the BankTeller resource entities, were available on a continuous basis. In order to add realism, we choose to model the lunch breaks taken by the tellers. Assuming that each unit of the “simulation clock” corresponds to one minute of time and the simulation runs from time 0 to time 540, each run simulates one business day (8:00 a.m. to 5:00 p.m.). Two of the tellers take their lunch break from noon to 1:00 p.m. and the other from 1:00 p.m.to 2:00 p.m. This means that all three tellers are on duty for the first four hours (8:00 a.m. to noon), only one for the next hour (noon to 1:00 p.m.), two for the hour following (1:00 p.m. to 2:00 p.m.), and all three again for the final three hours of the day (2:00 p.m. to 5:00 p.m.). We can model this in the Resource Agenda block, whose block properties box is shown in Figure 7.

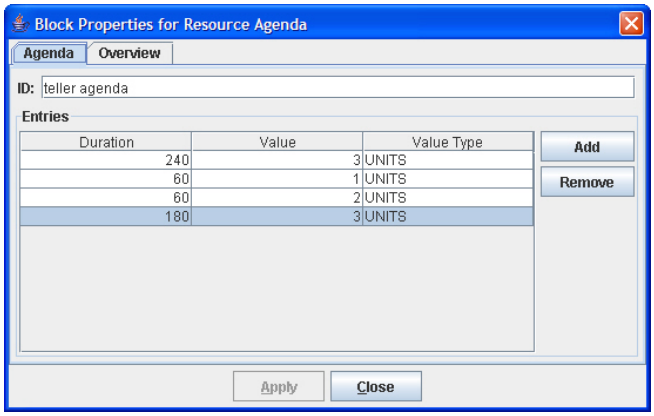

Figure 7. Details of the Resource Agenda Block for the Bank Lobby Model

The role of the Resource Agenda block is to specify what resource adjustments to make, when they are to be made, and the duration of each adjustment. The resource adjustments discussed above are clearly reflected in the details of the agenda named “teller agenda” in this block. Note that in addition to the “Duration” and “Value” columns, this Block Properties dialog box includes a column named “Value Type.” In this model the value in this column is “UNITS,” indicating that we wish to adjust the number of resource units available. If you prefer you can choose the value “STATE” for this column, indicating that the agenda is used to change the state of the resource entity—between “Functional,” “Failed,” “Maintenance,” and “Offlined” (indicated using the “Value” column). Adjustments of both types, units and state, can occur in the same resource agenda.

Once we have defined a resource agenda we must specify how it is to be implemented when the simulation model is run. This is the function of the Resource Scheduler block, with block properties as shown in Figure 8.

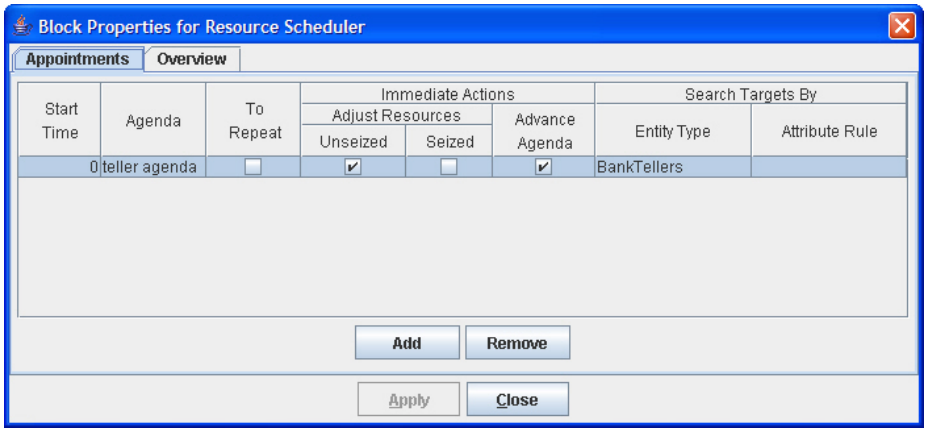

Figure 8. Details of the Resource Scheduler Block for the Bank Lobby Model

The Resource Scheduler block can accept several different resource agendas as input; for each an “InAgenda” input port is established and we connect the corresponding Resource Agenda block to it. The Resource Scheduler performs the adjustments sequentially and as directed for each listed agenda. In order to implement the each agenda correctly, the Resource Scheduler block also resolves several critical issues regarding the implementation of a resource agenda:

- When should the agenda be started?
- What resources should be adjusted?
- Should the agenda be repeated once it has been executed?
- Should the resource adjustments be made on unseized resources, seized resources, or both? (Are resource adjustments disruptive or preemptive for seized resources?)
- How should the agenda proceed to the next planned adjustment?

The first three issues are the easiest to address. In the Block Properties dialog box for the Resource Scheduler (Figure 8), the “Start Time” value of “0” indicates that the “teller agenda” (indicated in the “Agenda” column) starts when the simulation run starts, meaning that the timing of its resource adjustments is relative to the start of the simulation run. The “Entity Type” value indicates that we are adjusting the BankTeller resource entities, and the unchecked box in the “To Repeat” column indicates that this agenda is to be implemented only once.

Resolving the issue of adjusting seized or unseized resources (or both) in the model chiefly is a question of imitating what occurs in the real system being simulated. For the bank lobby, the question becomes “When a teller’s lunch break arrives, does the teller begin the break even if he or she is helping a customer at that point or does he or she conclude the transactions and then begin the break?” If the break occurs immediately (even if the teller is occupied with a customer), then the resource adjustment should be made on seized resources. If instead the teller waits until the customer has finished (until the BankTeller resource entity is released and returns to the Teller Pool), then adjustments should not be made on seized resources. In this model we have indicated that only idle tellers (unseized BankTeller resource entities) should begin their lunch breaks by checking the “Unseized” box under “Adjust Resources” in Figure 9 and not checking the “Seized” box.

The last question listed above can become important if, as in this model, we do not check the “Seized” box and consequently must wait for resources to become unseized before adjusting them. The adjustment (the lunch break in the bank lobby model) is thus delayed; what should be done about the rest of the agenda? Should it be similarly delayed or should the next adjustment (here, the end of the lunch break for some tellers) occur as planned? If you check the “Advance Agenda” box in Figure 9, then subsequent adjustments will occur as planned; otherwise the next adjustment will be delayed. Note that if several adjustments are delayed, the overall delay in the agenda could become significant since each delay is passed forward to the next adjustment in cumulative fashion. In the bank lobby model this does not happen since we check the “Advance Agenda” box. This does indicate that if a teller is occupied at the start of the lunch break and takes, for example, three minutes to finish with the customer, his or her lunch break is shortened by three minutes.

COLLECTING RESOURCE ENTITY STATISTICS

The last new aspect of the revised bank lobby model is the Resource Stats Collector block. Its use is relatively simple here, but this block offers great flexibility in organizing, monitoring, and collecting statistics on resource entities that at any time during the simulation run might be scattered throughout the model.

In the block properties for the Resource Stats collector block, shown in Figure 9, the “Groups” tab enables you to select the resource entities on which you want to collect statistics.

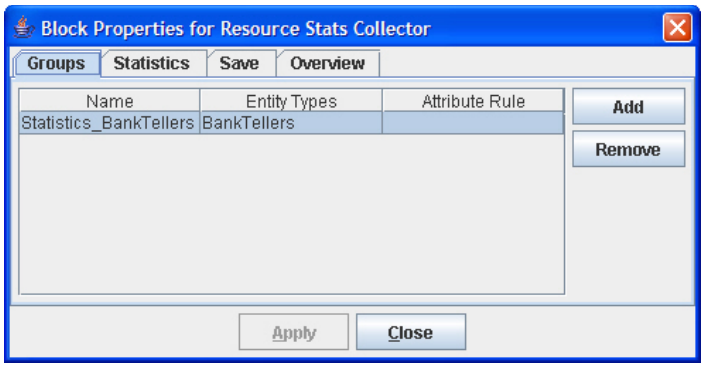

Figure 9. Groups Tab for the Resource Stats Collector Block

For this model BankTellers is entered in the “Entity Types” column. Note that if you wish you can also use the “Attribute Rule” column to specify a subset of the chosen entity type on which to gather statistics. This rule could specify one or a combination of values for one or more attributes that are common to the specified entity type.

Once you’ve defined your targeted group or groups of resource entities, you need to specify which statistics to collect. This is the function of the “Statistics” tab, shown in Figure 10.

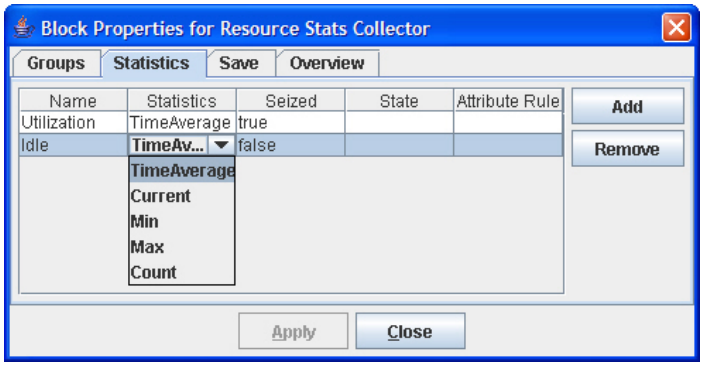

Figure 10. Statistics Tab for the Resource Stats Collector Block

Here we define two statistics, Utilization and Idle, both of which are time averages. Figure 10 illustrates, though, that several types of statistics are available. Utilization and Idle are defined as the average percentage of the time during which the resource entity (BankTellers) are seized or unseized, respectively. As with the Groups tab, more

functionality is available. The “State” column enables you to specify a specific state for the resource entity and the “Attribute Rule” column enables you to define a subset of resource entities based on attribute values, in each case indicating that the specified statistics should be computed only for qualifying resource entities. Using the “Save” tab, you can specify the directory to which these statistics are saved.

CONCLUSION

Discrete event simulation is a modeling and analysis approach that is noted for its ability to capture the random variation and behavior that is critical in understanding and improving the operations of real-world systems. SAS Simulation Studio, a graphical application, provides you with the tools to apply discrete event simulation modeling and analysis methods to a broad range of systems and problems and create accurate, detailed, and flexible models in all cases.

Accurate, detailed modeling is especially important for resources, which are so influential in discrete event simulation models and in the systems that the models represent. Fortunately, SAS Simulation Studio provides an excellent set of modeling capabilities that enable you to capture in detail the behavior of both stationary and mobile resources, contributing to discrete event simulation models that yield richer and more useful performance data. Many more detailed and more advanced capabilities are available than could be covered in this paper; please consult the *SAS Simulation Studio User’s Guide* for details.

REFERENCES

SAS Institute Inc. 2009, *SAS Simulation Studio 1.5: User’s Guide.* Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENTS

Thanks to Emily Lada and Phillip Meanor of the SAS/OR R&D staff for their many valuable contributions to the editing of this paper.

RECOMMENDED READING

Hughes, E., Meanor, P., Lada, E., Chen, H., “Exploring System Performance with SAS Simulation Studio” (Paper 317-2009)*, Proceedings of the SAS Global Forum 2009 Conference. Cary, NC: SAS Institute Inc.*

CONTACT INFORMATION

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

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