Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

MICRO SAINT SHARP SIMULATION SOFTWARE

Wendy K. Bloechle
Daniel Schunk

Micro Analysis & Design
4949 Pearl East Circle, Ste. 300
Boulder, CO 80301, U.S.A.

ABSTRACT

For the past nineteen years, Micro Saint simulation software has been helping the military and other commercial companies answer questions on how to improve performance and utilization for their various processes. Recently, Micro Saint has been redesigned to be faster, modular and more powerful. Because these changes represent such a major change from the original Micro Saint, we are releasing a brand new tool called Micro Saint Sharp. Micro Saint Sharp is still a general purpose tool that can be used to provide solutions ranging from queuing problems involving hospital waiting rooms to complex human decision processes involving future command and control systems. This paper will provide an overview of Micro Saint Sharp and present some of its new modeling capabilities.

1 INTRODUCTION

Discrete event simulation has been a standard technique in the analysis of manufacturing systems for years. Attention to the value of simulation in evaluating alternative strategies to reduce costs has also increased in other industries as a result of competitive pressure and rising costs. Simulation is now being used to evaluate and improve efficiency in a myriad of areas, including: process definition, quality measurement and control, process re-design, employee workload, safety and productivity. With decision-makers applying simulation technology to a wider variety of problems, the need for general purpose simulation tools that are capable of addressing all these needs has increased. Micro Saint Sharp, a task network based modeling tool, is recognized to be an efficient and cost-effective tool for simulating the complexities of systems within a variety of industries ranging from the military to health care. Problems being analyzed within these industries range from process control and resource utilization, to military maintenance procedures and human performance.

The purpose of this paper is to provide a basic understanding of the principles of modeling with Micro Saint Sharp. Micro Saint Sharp does not use the terminology or graphic representations of a specific industry. If the system can be drawn as a flow chart, then you can build a model of your process in Micro Saint Sharp.

The degree of model complexity is flexible. A simple, functional model can be built just by drawing a network diagram and filling in the task timing information. Also a more complex model that includes dynamically changing variables, probabilistic and tactical branching logic, sorted queues, conditional task execution, and extensive data collection can be built in Micro Saint Sharp. Micro Saint Sharp includes a fully functional programming language. This sophisticated programming language will make it more efficient to write code especially for large, complex models.

Whether the model is simple or complex, the process of running the model and generating statistics and graphs from the collected data is relatively simple. The user simply selects the execution settings and the variable data he/she wants collected. Micro Saint Sharp symbolically animates the network diagram as it executes the model, using a random number seed the user provides to generate task times and routing choices specific to the current run. After running the model, the user can select statistics charts, scatter plots, line or step bar graphs, bar charts, and frequency distributions to analyze any data you collected. In addition, Micro Saint Sharp automatically collects queue data.

One of the key features in Micro Saint Sharp is the customizable development environment. While the software is provided with a default setting when the program is initially opened, the user can move the palettes and tools around to create a display that will be efficient for his/her use.

Animation development is provided by a tool called Animator. A major drawback of some simulation software packages is that there is only one view of the process that is available – either the flow chart or the animation. This can make model debugging time consuming and difficult for the simulation modeler. Micro Saint Sharp offers two views of your process. With the network diagram view, you can animate the process flow chart and change task ovals to any appearance you choose. With the Animator module, users

can show a more realistic picture of the process with moving components. Animator provides the capability for charts and graphs, text, and images to all be displayed while the model is running on one screen. In addition, once the animation is created, users will be able to send the animation portion of the model to others and to view it through the Animator player without having to own Micro Saint Sharp.

Micro Saint Sharp was developed specifically with the goal of increasing model execution speed. While users will still be able to control the speed of model execution, they now can run the model with all of the interfaces turned off. This mode will increase execution speed by a factor of 10. So when the goal is to have results as quickly as possible, this mode will allow simulation modelers to turn off the visual components.

Modularity is also unique to Micro Saint Sharp. Many simulation tools come with all the components in one package whether you need them or not. Because of Micro Saint Sharp's new plug-in interface and object oriented model development, it is easier to do integration. Not only with other external software applications but with its own individual tools. The optional tools available with Micro Saint Sharp include optimization using OptQuest, animation using Animator, and interoperability using COM Services.

In the following sections, an overview of how to build a model in Micro Saint is provided.

2 MODELING ELEMENTS

Micro Saint Sharp uses a methodology known as task network modeling. Activities are represented in a diagram as nodes and the arrows between the nodes represent the sequence in which the activities are performed. A simple task network is shown in Figure 1.

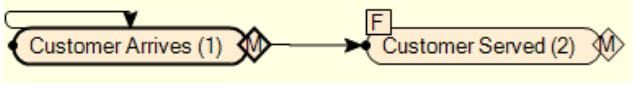

Figure 1: Simple Task Network

This approach allows users to develop models using the same techniques they would use to define a flow diagram of the activity. Each activity, whether it is a human activity or a system activity, is defined using the same method.

A Micro Saint Sharp model is composed of "networks" which may be a sequence of tasks to be performed by a human, a series of processes that define an organization, or a machine in a manufacturing plant. Networks are composed of either lower-level networks or "tasks." Tasks represent the lowest level in the model and have specific parameters (timing information, conditions for execution, beginning and ending effects).

3 MODEL DEVELOPMENT

The process of building a Micro Saint Sharp model involves two separate but interrelated steps. First, the user must define the structure of the task network. This is done by either dragging a task or network object from the palette onto the task network diagram or by right clicking on the network diagram and selecting the menu item "Add Task" (see Figure 2).

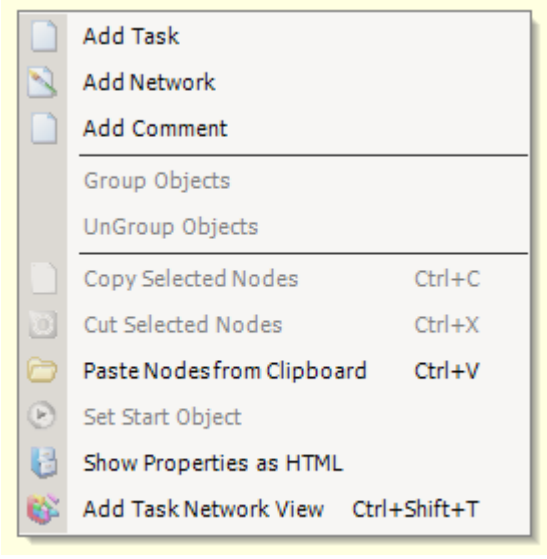

Figure 2: Task Network Context Menu

Second is to define the objects in the network. Micro Saint Sharp uses the windows standard "point and click" approach to define network objects. Using the mouse to "double-click" on an object will open it so that information specific to the object may be entered. Micro Saint Sharp also has a "Properties" window allowing the user to edit any object's properties after it has been selected. The following section explains the task parameters in more detail.

3.1 Task Timing Information

The task "mean time" is the average time that a task takes to complete once it has begun executing. For example, if the task represents a human activity such as "transfer patient to recovery room," then the mean time to execute is the average time that it takes to perform the task. If the task represents a machine in a manufacturing process, then the mean time to execute is the average processing time for the machine. In many cases the execution time is not constant, rather, the elapsed time falls within a range of values that can be represented by a time distribution. Micro Saint supports more than 21 distribution types including normal, rectangular, exponential, gamma, Weibull, Poisson and others. In addition, users may enter their own parameters to control the spread of the distribution.

Alternatively, the mean time may be determined by the current state of the system or by an attribute of the process itself. In human performance modeling, the mean time to perform a task may be influenced by such conditions as how long the human has been working, the skill level of the human, or the current workload. In an insurance claim processing model, the type of claim or the location of the client may determine the time it takes to process a claim.

In Micro Saint Sharp any information that is used by the engine to make calculations must be "returned" to the engine. This is done in the mean time and standard deviation description boxes with the code "return" followed by either a number or variable representing the mean time or standard deviation (see Figure 3). Micro Saint Sharp will use this number in its calculation of the task duration time.

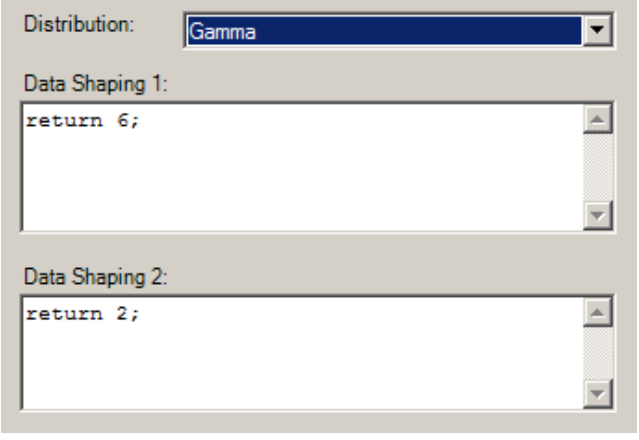

Figure 3: Mean Time and Std Dev Return Code

3.2 Conditions for Execution

Often, there are situations where a task cannot begin executing until certain conditions are met. A customer cannot make a transaction at a bank, even though the queue is empty, until a bank teller is available. A task may have resource requirements or other constraints (i.e., time of day, part type) that dictate when the task may begin executing. In Micro Saint Sharp, a Boolean (logical) expression is evaluated in the "release condition" field to control the execution of tasks. The release condition description box is another box which needs the "return" code. This return expression is then used to evaluate whether a task can be started or not. Entities moving through the network, such as patients, parts, or claim forms, cannot be released into a task for processing until the release conditions for the task have been met.

3.3 Beginning/Ending Effects

The current state of the system may change when a task begins or ends. For example, when a machine begins processing a part it becomes "busy" and is not available to another part until it has finished. The user would define the following expressions in Micro Saint to define this condition:

- Release Condition: if busy == 0 then return 1 else return 0; This keeps an entity (e.g. part, patient, etc.) from moving into the task when the task is "busy" processing another entity.
- Beginning Effect: busy = 1; This sets the busy flag to TRUE so that the next entity cannot enter the task. As long as the task is executing, the busy flag will remain equal to "1".
- Ending Effect: busy = 0; When the task finishes executing, the ending effect is evaluated and the busy flag is set to 0. Now, when the release condition is evaluated, the condition will be true and the next entity can enter the task.

This relationship between the release condition and the beginning and ending effects provides a general, yet powerful mechanism for users to define complex behaviors within the system they are modeling. Users may define variables that are specific to their system and manipulate the value of the variable as needed so that they can accurately represent their system. They do not have to compromise the accuracy of their model by relying on predefined "blocks" within the modeling tool nor do they have to learn a complex programming language in order to obtain the level of control required.

3.4 Task Sequencing

Task sequencing is defined by clicking on the diamond after a task and dragging with the mouse from the first task to the following task(s). Users then enter the conditions that control the branching; this includes tasks with just one exiting path or multiple exiting paths. Micro Saint provides the following decision types to ensure that all real-world situations may be represented in the model:

- Probabilistic: The following task conditions are evaluated and the next task to execute is determined by the relative probabilities of all tasks listed. Only one of the following tasks will be executed with probabilistic decisions.
- Multiple: The following task conditions are evaluated and all of the tasks whose conditions evaluate to non-zero will execute.
- Tactical: The following task conditions are evaluated and the next task to execute is the task whose condition evaluates to the highest value.

Variables and algebraic expressions can be used in the branching logic and the value of the variables can be changed by conditions in the model. This gives the user complete control and manipulation of the network flow.

All of these features provide an environment for the model developer that is easy to learn and easy to use. Once the basic concepts are understood, any system or process can be modeled using Micro Saint. In addition, users can build models at any level of complexity.

4 VARIABLES

Variables allow a user to change any aspect of a simulation model. A user defines variables based upon what information he/she is trying to get out of a model. Variables are defined by the user by editing variable properties. These properties can be accessed by clicking on a variable in the variable list and editing that variable via the properties window (see Figure 4). Micro Saint Sharp supports the following variable types.

- Integer: Any number that can be counted in whole numbers (e.g. people, parts)
- Floating Point: Any number that could have a decimal point (e.g. time, temperature)
- String: Words and phrases
- Boolean: true and false logic statements
- Entity: Points to a specific entity inside the simulation model.

Variables can also be defined inside the model locally by declaring the variable type and name. Micro Saint Sharp also supports variables specific to entities called entity attributes. This allows the user to specify specific attributes of every entity inside a Micro Saint Sharp model. These specific entity attributes could then be used for data collection or to affect how an entity could flow through the model.

5 FUNCTIONS

Micro Saint Sharp has two types of functions: built-in functions, which are accessible from all models, and custom functions, which the user defines within a particular model. When Micro Saint Sharp encounters the function, it executes the function and returns a value that can be used in the expression. Custom functions are particularly useful for calculations or procedures that the user wants to execute at more than one place in a model. For example, a function to calculate machine utilization might be created and used to calculate utilization for various machines.

6 ANALYSIS AND RESULTS

People build models to provide insight to, or to answer specific questions about, a system or process. Some information can be gained by watching the Micro Saint Sharp model run. Micro Saint Sharp's symbolic animation capability provides an animated view of the network diagram as the model is running. Users can watch as entities flow through the network or wait in queues before being processed. This type of animation is particularly useful in debugging the model.

Sometimes it is sufficient to save the state of the system at the end of the run. However, in order to gain insight into the dynamic aspects of the system, users can "take snapshots" of the model variables at any time during the run. These "snapshots" of data can be analyzed by importing them into another statistical analysis package. In addition, data can be collected at any time during the model run. Micro Saint Sharp automatically collects data for every queue in a model. In addition, task, trace and resource data files can be collected automatically.

Micro Saint Sharp includes a real time graphing capability. This graphing capability allows the user to watch data be dynamically graphed onto any type of chart. These charts can then be printed or saved as separate files for later analysis (see Figure 4). Using the insights gained from the results of the simulation analysis, users can assess the relative merits of alternative solutions. Additionally, users can predict the impact of these solutions which subsequently leads to a better understanding of the costs and benefits.

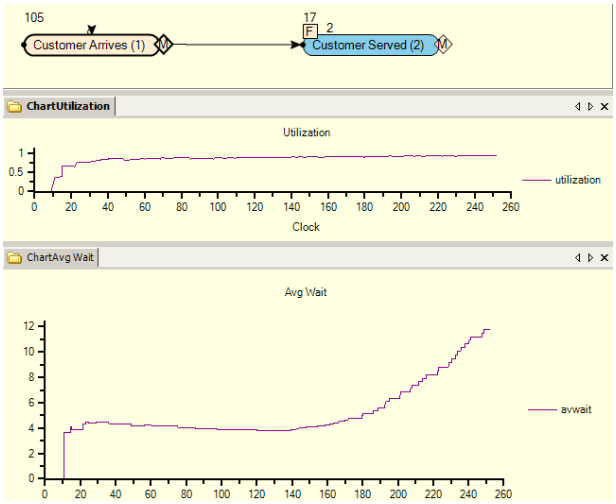

Figure 4: Network Animation and Charts

Micro Saint Sharp also has an image-based animation capability called Animator. This allows custom animations of the model to be built. The background scene can be a three-dimensional diagram from a CAD package, a digitized diagram of a factory floor, or a bitmap from any drawing package. The creation and movement of the animation is driven by events that take place in the model. Animator allows descriptions of the model to be shown while the model is executing and is an extremely valuable tool for presenting the model. It can also be run in real time. Using Animator, the user can have annotations, labels, plots, graphs and images all on the same screen. Images can be scaled and rotated with ease. After a model has been executed, the animation run can be saved and played later via the Animator player.

7 BANK TELLER MODEL

The bank teller model represents a simple, classic queuing simulation model. In this model, the bank opens at 9am and people are no longer admitted into the bank after 5pm. During lunchtime, a larger number of people come into the

bank to perform their transactions. There are currently two bank tellers on staff during the day.

7.1 Define Variables

The variables used in this model can be broken out into two different categories: model variables, variables that define how the model itself will run and data collection variables, variables used specifically for data collection.

The model variables are:

- Rate: defines the current rate of arrival of bank customers
- Tellers: defines the current number of available tellers at any given moment during a model run

The data collection variables are:

- Waittotal: used to calculate the total wait time of every entity in the model
- Avwait: used to calculate the average wait time of the entities in the model
- Maxwait: used to calculate the maximum time an entity had to wait in the model
- Servetotal: used to calculate the total service time of every entity in the model
- Utilization: used to calculated the utilization by teller
- Maxtellers: defines the maximum number of tellers available, used to calculate the utilization by teller
- Waited: used to calculate wait time by customer
- Served: used to calculate serve time by customer

7.2 Define Task Network

The task network consists of two tasks that every entity will perform: arrive at the bank and be served at the bank. The entity in this model is the bank customer. The task network is shown in Figure 5.

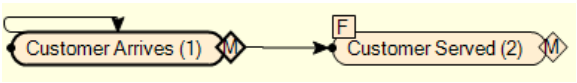

Figure 5: Bank Teller Task Network

Entity arrivals are defined by the Rate variable and will account for the possibility of entities entering the system at different rates. Entities are also tagged after they arrive into the system in order to make each one a unique entity in the system (see Figure 6).

When an entity finishes its arrival task, the next arrival needs to be scheduled unless the bank is not permitting any more people to arrive. This decision type is described by multiple paths: one for the next arrival and one for the customer to get in line to be served. When the simulation clock is past the standard eight hour work day, no customers will be allowed to enter the bank (see Figure 7).

Figure 6: Entity Tag Code

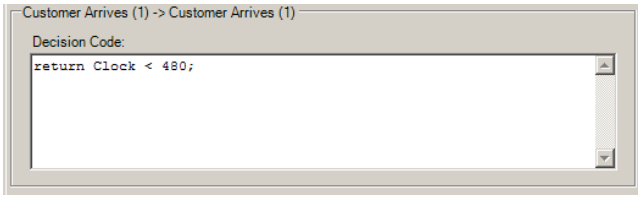

Figure 7: Customers will Stop Arriving into the Bank after 8 Hours

After a customer has arrived at the bank, he/she needs to get in line to be served. If there is no line, then the customer is immediately served. When the customer starts to be served, data is collected so that teller utilization can be calculated (see Figure 8).

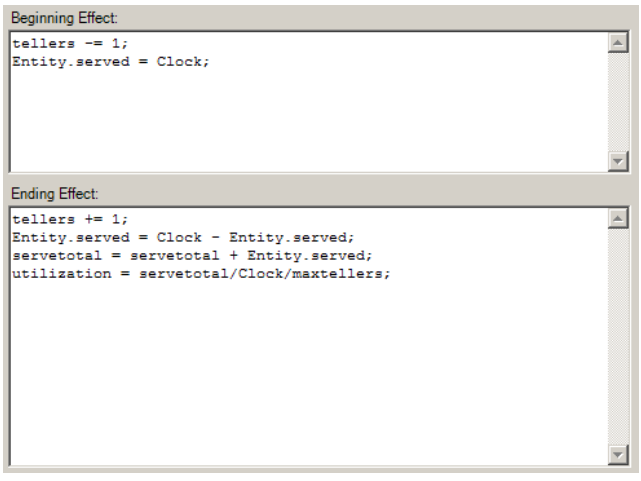

Figure 8: Utilization Calculation Code

If all the tellers are busy, then the customer needs to wait in line. The line will be a first in, first out (FIFO) queue. Data collection is useful here in order to analyze customer satisfaction. Average customer wait time and maximum customer wait time will be calculated.

7.3 Define Scenario Events

The scenario of the model will need to define all the initializing factors of the model as well as account for the rate of arrival change of the customers at noontime. A series of

scenarios will define that there are two tellers on duty and that the rate of arrival will be one customer every three minutes. At eleven (or two hours into the simulation) the rate of arrival will increase to one customer every two minutes, and then at one pm (four hours into the simulation) the rate of arrival will return to one customer every three minutes.

7.4 Define Data Collection

For this model, two types of data collection will take place, visual and analytical. For the visual data collection, a real time chart will be developed. This chart will show average wait time and utilization over time. Once a chart is added to the model, two data series points will need to be added, one for utilization and one for average wait time. A chart will appear upon model execution and update during the model run.

Snapshots are added to the model in order to get results data after model execution. The snapshots for this model will record the final values of the average wait time (Avwait), the maximum wait time (Maxwait), the time the simulation ended (Clock), and the utilization by tellers (utilization).

7.5 Data Analysis

After the model runs and the data has been collected, the user can analyze the optimal number of tellers to schedule throughout the day. It is interesting to note in the graphs how much of an effect the noontime rush has on wait times.

It is also interesting to note how much of a difference there is in both wait times and utilizations between having four tellers on duty versus two or three. Another item to analyze would be adding another teller part-time during the lunch hour.

8 SUMMARY

Micro Saint Sharp represents a breakthrough in simulation technology. With the increased speed and flexibility that Micro Saint Sharp offers, users will be able to get the answers to their questions quickly. These answers will help lead to reduced costs, improved performance, time savings and better customer service.

AUTHOR BIOGRAPHIES

WENDY K. BLOECHLE is the Marketing Director at Micro Analysis & Design. Her primary responsibilities are in supporting new business development for the company's simulation products and services. She received her B.S. in Industrial Engineering from the University of Illinois and her M.B.A. from the University of Colorado. Her email address is <wbloechle@maad.com>.

DANIEL SCHUNK is an Industrial Engineer for Micro Analysis and Design. He is the lead Technical Support Engineer for Micro Saint Sharp and is the main instructor for the training class. He has a Bachelor of Science in Industrial Engineering from Purdue University. His email address is <dschunk@maad.com>.