ABSTRACT
The tool called simulation has long been used to study the behavior of a system without actually having to build that system. New with release 6.11 of SAS/OR® is QSIM, a GUI-based application for constructing and analyzing discrete-event simulation models. This paper explores a number of traditional and nontraditional examples of using simulation and the QSIM application to solve a wide variety of business problems. The approach is practical and the material is appropriate for users with widely varying levels of simulation expertise.

INTRODUCTION
Simulation is defined as "an experiment in which we attempt to understand how something will behave in reality by mimicking its behavior in an artificial environment that approximates reality as closely as possible" (Shogan 1988, p.749). Simulation provides companies with a cost-effective means of testing the feasibility of strategies and projects prior to implementation, thereby substantially decreasing the risk of failure. Although the theoretical framework underlying simulation is complex and highly quantitative, use of simulation to effectively analyze options is not limited to academics and otherwise technical individuals. Today, tools within the SAS® System provide a means by which those with limited prior experience can easily exploit the power and flexibility of simulation-based analysis. The availability of a user-oriented means of simulation is timely in this era of corporate process reengineering.

This paper will explore, on a practical level, the process of simulation, introduce the QSIM discrete-event queuing environment, and provide three short applications of simulation to real-world problems.

A STEP-BY-STEP APPROACH
As with most analytic approaches, a disciplined, step-by-step process can be defined for simulation studies. Although somewhat abbreviated, the following approach is based on the steps outlined by Law (1986).

1. Formulate the problem. The purpose of this step is to gain a firm understanding of what is to be modeled, and what needs to be learned. Devise a list of questions that it will be the goal of the simulation to answer.

2. Collect data and define a model. In the case of an existing system, this step requires detailed analysis of how the system currently works, and collection of data about each component of the system. In the case of a non-existent system, effort must be spent understanding how the system will work, and data about each system component must be estimated. This step requires critical subjective judgments as to the level of detail that is included in the model.

3. Construct a computer model and verify it. In this step, the conceptual model defined in step 2 is built using a computer simulation facility. Preliminary data are fed through the model to help identify any errors in the logic of its construction, and as reassurance that the simulation model generates output that is in line with what is expected.

4. Experiment on the system. Once an acceptable computer model is constructed, the behavior of the system can be studied. Experiments are conducted to evaluate the impact of changes in resource levels, or other model alterations. Ultimately, experimentation and analysis of the resulting output should answer the questions identified during the problem formulation phase.

The SAS System includes a robust set of tools for aiding in each of the four phases of the simulation analysis process. The descriptive statistical procedures that have been an integral part of the SAS System for many years are clearly of use in the initial phases. New is the QSIM discrete-event simulation application to assist in building the computerized representation of the model, and to analyze the results of experimentation.

THE QSIM DISCRETE EVENT SIMULATION APPLICATION
The QSIM application is a SAS application for modeling and analyzing queuing systems using discrete event simulation. These models are used in a wide variety of scenarios that might be encountered in network and telecommunications systems, manufacturing systems, transportation networks, and in a host of service-based systems.

The application has a graphical user interface that requires no programming and provides all the tools for building, executing, and analyzing discrete-event simulation models.

With the QSIM application, you can build a model of a queuing system, control the simulation of the model, and produce summary statistics from the simulation sample path from within the application. You can save the model and the sample path in SAS data sets for reuse and further analysis. The application is designed to simplify model building by encouraging the construction of hierarchical models based on user-built model components that can be stored, shared, and replicated easily.

Figure 1. The QSIM Primary Simulation Window.
Statistics, Data Analysis, and Modeling

The primary QSIM simulation window is shown in Figure 1. The window has two panels: the toolbar panel on the left, and the model panel on the right (a simple queuing model has been built within the model panel). You build the model in the model panel, using a mouse to select model components from a popup menu or from the toolbar. These model components are connected by arcs to produce a directed network representation of the model. You can change parameters and other properties of model components via popup menus from the components.

Although there are many components that can comprise a given simulation model, very few will figure prominently into this paper. Definition of the most common model components is appropriate at this point:

**Sampler.** A sampler initiates transactions with a prescribed inter-arrival time. The inter-arrival time could be very simple such as every 5 minutes, or it could be non-constant and follow a distribution. An example of a transaction is a shopper in a grocery store who is ready to check out.

**Queue.** A queue holds a transaction while it waits for access to a server. Queues may follow a number of different strategies for releasing transactions to servers. A first-in-first-out (FIFO) queue is similar to a line at the grocery store's check out stand.

**Server.** A server holds a transaction as it is served. The service time can be a constant time value, a sample from a distribution, or a value read from a SAS data set. An example of a server is the checkout clerk at a grocery store.

Referring again to Figure 1, it is easy to see how these three components combine to form the representation of a simple (M/M/1) queuing system. On the left side of the model panel is a sampler generating transactions. The transactions flow toward the right into the queue, and finally passed on to the server.

**Examples**

Having described the QSIM application, and having defined a few basic simulation components, the best way to proceed in understanding the ease and power of the application is through example. The remainder of this paper is devoted to two examples of QSIM in action. The first example is a simple production line for constructing refrigerators from parts. The second example is a bank teller queue which incorporates multiple servers and customers who refuse to wait in line because it is too long (reneging).

**Example 1. Refrigerator Assembly**

A manufacturing company wants to study the feasibility of a new refrigerator assembly operation. Refrigerator cabinets are assembled in one area of the factory, and doors in another. Ultimately, the doors and cabinets are joined together to form the final product. By experimenting with varying levels of resources, they hope to design an assembly operation that flows smoothly and gets completed products out the door.

Figure 3 is a graphical representation of the refrigerator assembly operation. The model components and the assumptions governing each are described below:

1. Handles, doors, frames and floors are the samplers in the model. There is an infinite supply of each, so transactions are passed on to the assembly operations as quickly as those operations are able to take them.

2. Door assembly is comprised of a queue and a server. The queue is FIFO and can hold up to 50 waiting transactions at a time. The server has a service time that follows an exponential distribution with mean 5. In Figure 3, the queue and server have been combined into a composite component called "assembly." The ability to create composites is one of the object-oriented features of QSIM.
3. Cabinet assembly is a composite identical to the door assembly. In this case, however, the server follows an exponential distribution with mean .63. Apparently it takes longer to assemble cabinets than doors.

4. Final assembly is, once again, a composite incorporating both a queue and a server. In this case the service time is exponentially distributed with mean .35.

5. The final items that require explanation in Figure 3 are the strip charts to the right. These charts collect information on refrigerator throughput, the door assembly buffer, and the cabinet assembly buffer from top to bottom, respectively.

One interesting feature of this example is the use of animation. Animation is a toggle that can be turned on or off. When in the on position, transactions can visually be seen as they flow through the system. This feature is particularly useful when verifying that your computer model behaves as expected. By default, transactions look like tiny black boxes, however this default style can be overridden with icon representations that better resemble reality. This example uses icon representations of handles, doors, frames, floors, partially assembled refrigerators, and completed refrigerators. Note that enabling animation slows the simulation dramatically, so when a large number of transactions, or a long period of time is being simulated, its best to turn animation off.

Once the graphical representation of the model is complete, the simulation control panel is used to start the simulation. When this simulation is executed, it becomes clear very quickly that there is excess capacity in door assembly. The evidence of this comes from the strip chart that is monitoring the door assembly buffer. Doors cannot be passed off to final assembly, because the corresponding cabinets are taking too long to produce. Therefore, this assembly line design performs poorly and its resources should be realignd.

Ideally, resources would be reallocated so that door assembly and cabinet assembly take closer to the same amount of time. This might be accomplished simply by shifting manpower (and altering the server times). Or it may mean adding an additional cabinet assembly operation (which is only as difficult as copying the existing one). Either way, the model can easily be updated within QSIM and the new results can be examined until a more optimal refrigerator assembly configuration is found.

Example 2. Bank Tellers

A bank wants to study the flow of customers through its teller line, both as a means of understanding the efficiency of its operations, and as part of an initiative to improve customer satisfaction. The bank knows that if too many tellers are on duty, it is wasting money. It also knows that if a customer arrives and the line is too long, customers will leave unsatisfied.

Figure 4 contains the QSIM representation of this problem. The key model components and assumptions are as follows:

1. Arrival is a sampler that generates transactions. In the case of this model, a transaction is a bank customer walking in the door. Assume that customer inter-arrivals follow an exponential distribution with mean of 2.04 minutes.

2. Once a customer has entered the bank, the first thing he or she does is look at the length of the line. As an initial as-

sumption let us say that a customer will join the line if there are five or fewer individuals already in line — if there are more than five people waiting, the customer will turn around and leave. This is modeled with a QSIM component called a splitter. The splitter component is able to look at the length attribute of the service queue, and split transactions down different paths based on the value of that attribute. Thus, some transactions will join the service queue, while others will be routed downward to the renge queue.

3. The renge queue is really only a holding place for accumulating renge transactions. The probe chart below this queue shows the cumulative count of transactions in this queue across time — this is a means of monitoring renge activity.

4. The service queue is a FIFO queue leading to the tellers. Its practical capacity is five, as transactions will be routed to the renge queue rather than the service queue if there are already five people in the queue. Above the service queue is a probe that monitors the size of this queue.

5. Finally, tellers is a seven unit multi-server component. The n unit multi-server component is equivalent to n individual identical servers, but is much simpler to create and maintain. Assume that each server has a service time that is exponentially distributed with mean 2.99 minutes. The strip chart beneath this component monitors the age of completed transactions, from the time they enter the door to the time their service is complete.

Figure 4. Bank Teller Simulation.

When the simulation is run, it quickly becomes clear that customer service goals are not being met, because the renge rate is very high. Analysis beyond that which is provided by the on-screen charts could be performed, by opting to collect the simulation into a SAS data set. The simulation data could then be analyzed using any of the SAS Systems many analytic tools.

In the case of a high renge rate, the bank has a couple of options: the first is to add tellers; the second is to see if the service time mean can be reduced by streamlining service processes. Using QSIM the bank can study both options, and work out a solution that improves service the most for the least cost.
CONCLUSION

As we have seen, simulation can be a powerful tool for addressing a variety of business problems. Although the examples in this paper have provided only a narrow glimpse into the power of the QSIM application, the power and flexibility of the tool should be clear. The design of the application makes model building simple for novice users, while a broad array of components and functions makes it appropriate for expert users as well. The ability to create composite components greatly shortens development time when complex model fragments are used over and over.

ACCESSING THE EXAMPLE FILES

The examples in this paper were adapted from the sample simulations shipped with SAS/OR. The modified examples are accessible to you through the Internet.

The following files have been stored on SAS Institute's Internet gateway:

SimBus.bin — contains portable versions of the models used in the paper.

SimBus.txt — contains instructions for importing the simulation models contained in SimBus.bin.

You can download these files if you have access to the Internet. To download these files, connect to ftp.sas.com. Once you are connected, enter the following responses as prompted:

Name (ftp.sas.com:userid): anonymous
Password: your email address

All SUGI 21 files are stored in the following directory:

/pub/sugi21

There is one subdirectory for each paper that has ancillary files. Download the following file in /pub/sugi21 for a complete index of all files in /pub/sugi21:

README.index

The file README.index has a description of each directory. The description will contain the title of the paper and the directory name where the files are stored.

You should use a binary file transfer method when transferring SAS data sets, SAS catalogs, and other binary files.

REFERENCES

