Proceedings of the 1995 Winter Simulation Conference
ed. C. Alexopoulos, K. Kang, W. R. Lilegdon, and D. Goldsman

INTRODUCTION TO SIMULATION

Andrew F. Seila

Terry College of Business
The University of Georgia
Athens, Georgia 30602-6255, U.S.A.

ABSTRACT

The purpose of this tutorial is to introduce the basic concepts of simulation, specifically discrete event simulation. Reasons for using simulation as a tool for decision making are presented, along with the process of conducting a simulation study. Types of available simulation software and statistical problems and considerations in simulation are also reviewed.

1 WHAT IS SIMULATION?

Computer simulation is the focus of the Winter Simulation Conference. This tutorial presents the author's view of this subject and thus represents his experience and interests. Over the past several years, excellent introductory tutorials have appeared in the *Proceedings* (Thesen and Travis 1991, Shannon 1992, Gogg and Mott 1993, Pidd 1994). The reader is referred to these for alternative views of the subject of simulation.

The term "simulation" has been used to mean quite a number of things. Usually it refers to a realization of a representation of some larger, more complex activity. For example, engineers build simulations of physical systems such as a ship's flow through water. Aircraft pilots are trained on flight simulators which are physical devices that recreate the response of the craft to various actions the pilot might take, and allow the pilot to learn how to control the aircraft. Video games can be considered simulations. Physical systems such as manufacturing systems as well as more abstract systems such as computer networks can be simulated on a digital computer. All of these simulations use a *model* to represent the behavior of a system that may or may not exist and that is generally much larger, costlier and more complex than the model. The model may be physical, as in the cases of the aircraft simulator, or it may just be represented as a computer program, as in the case of the manufacturing and computer network simulations. In all cases, the key idea is that the simulation is an alternative *realization* that approximates the system, and in all cases the purpose of the simulation is to analyze and understand the system's behavior under various alternative actions or decisions.

A simulation in its simplest form is just a sampling experiment that is performed using a model. In all of the examples in the previous paragraph, we can think of the simulation as sampling the behavior of the model. In some cases, such as the aircraft simulation, the sampling is done continuously over time; in others, such as the manufacturing systems and computer networks, the sampling is done only at certain points in time. In this paper, the term simulation is used to refer to a sampling experiment that is performed on a digital computer, and also to the numerical, statistical and programming methodology associated with this activity.

1.1 Systems and Models

A *system* is a set of interacting components, or entities. Generally, the components work together to achieve some objective. Systems we commonly encounter include hospitals, airport check-in and boarding facilities, telecommunications systems such as the telephone network, highways and the criminal justice system. These systems are large and complex, and would be difficult and expensive to experiment with directly. A *model* is an abstract and simplified representation of a system. The model represents the most important system components, and the way in which they interact. A *stochastic* model is a model whose behavior cannot be predicted with certainty, but is subject to randomness. Most models that are analyzed using simulation are stochastic. The description of these models includes probability distributions for the values of variables that cannot be known with certainty. In manufacturing systems, for example, the times when machines will breakdown, or the lengths of time required to repair the machines are usually unknown. In computer communications systems, the number of packets that will be transmitted and exactly when they will be transmitted are also random. In both cases, the model must describe the probabilistic structure of these processes.

1.2 What Types of Systems can be Simulated?

Simulation has been used to analyze a very large variety of system. For evaluating the performance of manufacturing systems, simulation has come to be the dominant methodology, and special modeling tools and simulation software have been developed just for application to these systems. The conference has a special track just for manufacturing applications and another track for general applications. Other systems that have been simulated include transportation systems (railroad, naval, highways and air transportation), computer/communications systems (multiuser, multitasking computer systems, computer networks, telephone networks), military systems (warfare, logistics, personnel), agricultural systems (harvesting, logistics), public service systems (hospitals and other medical delivery systems, public health systems, emergency vehicle dispatch, police dispatch and other parts of the criminal justice system, hazardous waste management and nuclear power safety), and business processes (financial and other business transaction systems) among others. Simulation is a methodology that is not specific to any particular application areas, but can be applied to any system that can be modeled using the modeling concepts that we will discuss later.

2 WHY USE SIMULATION?

2.1 Simulation as a Tool for Decision-Making

All simulations are developed to determine system performance under alternative designs or environments, with the objective of optimally designing or operating the system. For example, we might want to determine if it is cost-effective to purchase a high-volume computer-controlled manufacturing machine. If the machine will eliminate a bottleneck and increase the production rate substantially, then we should buy it, but if it just transfers the bottleneck to another place, without increasing the overall production rate then we should not buy it. A simulation will allow us to estimate the production rate under both decisions without having to experiment with the real system.

2.2 Implementing and Evaluating Decisions

All systems have parameters that define the quantitative and qualitative characteristics of the system. These parameters can define, for example, the number of trunk lines in a communications system, the arrival rate for calls, the mean number of busy trunk lines, or the mean number of calls waiting. We can classify parameters into two classes: *input* parameters and *output* parameters. The number of trunk lines and the arrival rate of calls, which are examples of input parameters, must be specified in order to provide enough information for the simulation program to operate.

Decisions may be represented by the values of some input parameters. For example, if we set the number of trunk lines to 7, then we have decided to operate the system with 7 trunk lines. In other cases, decisions may be represented by system logic. For example, we may wish to compare two policies for equipment replacement or preventative maintenance. These policies must be specified within the logic of the system, not by just setting some parameter values. Thus, decisions may be either part of the model input or model structure.

Output parameters are values that are determined by the system and usually are used to specify the performance of the system. For example, the mean number of busy trunk lines tells us how heavily utilized the system is. Other output parameters might be the mean waiting time for calls entering the system, or the mean number of calls waiting in the system. All of these output parameters tell us somethig about how well the system is operating, and therefore are the "answers" that we seek from the model. When the simulation is running, it will produce data that can be used to estimate the values of these output parameters.

2.3 Simulation vs. Other Alternatives

An alternative to simulation is to use probability theory to compute the output parameters. Both are legitimate methods to analyze system performance, but there are fundamental and important differences between them. Mathematical analysis is limited to a small number of relatively simple systems; whereas, simulation can be used to analyze any system whose operation can be described in terms of a model. When the mathematical approach is used, parameters can usually be computed exactly, at least to the precision of the computer. Simulation, on the other hand, requires that we first collect data from the simulation run, then use the data to estimate the output parameter. Thus, the best answer we can get from a simulation is a confidence interval giving the likely range of values of the parameter.

2.4 Reasons for Using Simulation

Simulation is one of the most frequently used system analysis methods. There are a number of reasons for this. First, simulation can be used to analyze models of arbitrary complexity. The complexity of the model is limited only by the ability of the modeler and the methodology to represent the system's complexity, and the computer's capacity to load and run the simulation program. The investigator's primary interest might be to experiment with the system model in order to find the design that maximizes one or more performance measures, or simply to study the behavior of the system. Experimenting with the real system is often out of the question because of the cost to implement system changes, or the potential danger that could result from some policies being tested. Often the modeling effort is useful in itself. The process of model development requires the system to be studied and understood well. This study frequently uncovers problems that were unknown or not understood before. Relative to other methodologies, simulation can carry more credibility with decision makers. For example, an animation, which is a visual representation of the model, can be used to demonstrate that the model actually approximates system performance. Thus, a simualtion feels more "real" than other methods for system analysis. Finally, the same set of simulation methods can be used to analyze any stochastic system, regardless of structure or complexity. If a mathematical approach is used, it must usually be tailored to the particular system being modeled.

3 STEPS IN A SIMULATION STUDY

The process of using simulation to analyze a system can be divided into the following logical steps:

1. **Problem Statement and Objectives**. A clear, concise statement of the decision problem, or the reason for developing the simulation model, should be given first. The modeler should know what types of decisions are anticipated and what system is involved. These two items will dictate virtually every other aspect of the modeling and analysis process.

2. **Systems Analysis**. The modeler must thoroughly understand the operations of the system to be modeled. If the system exists, this can involve a careful study including observation of system operations and interviews with the persons managing the system. At this point, system components and their interactions should be identified and described as a prelude to the model building stage. All potential input parameters and random variables involved in the model should also be identified here.

3. **Analysis of Input Distributions**. Each random variable in the model must be examined, and the form of its distribution and the distribution parameters be determined.

4. **Model Building**. Here, we actually construct the model by deciding which details of the system to include in the model and which to exclude. The model must be a simplified representation of the system, but we want to include enough detail to provide a good approximation of the system behavior.

5. **Design and Coding of the Simulation Program**. Once the model has been specified, it must be implemented in the form of a computer program. The nature of this step will depend greatly upon the specific software used to implement the model. We will discuss the simulation software in a later section.

6. **Verification of the Simulation Program**. *Verification* is the process of making sure that the simulation program is a faithful representation of the intended model. This is basically a "debugging" process, but it can be complicated by the fact that the simulation program involves random variates whose values cannot be predicted in advance.

7. **Output Data Analysis Design**. Our ultimate plan is to use the simulation program to estimate one or (usually) more system performance measures, i.e., output parameters. Before this can be done, we have to decide exactly what data will be collected from the simulation run(s) and what statistical procedures will be used to compute the performance measures.

8. **Validation of the Model**. Before the model can be used for decision making, we must make sure that it adequately approximates the behavior of the intended real system. Model *validation* is the process of making sure that this is the case. Usually, this step involves collecting data from the real system as well as the simulation, and comparing the two to make sure they do not differ substantially. At the conclusion of this step,

the model should be usable as a tool to evaluate decisions concerning the system.

9. **Experimental Design**. In order to use the model to analyze decisions relating to the system, we must decide exactly what simulation runs will be done, using which parameter values and how long these runs will be. This, of course, depends upon what decisions are to be evaluated. The result of this step will be a design for the simulation production runs in the next step.

10. **Making Production Runs**. Once the experimental design has been set, the actual production runs can be performed to produce the output data. At this point, this step is usually a mechanical act. It is usually a good idea to store the data from the production runs on files so the runs will not have to be repeated if there is a change in the procedures used to analyze the data.

11. **Statistical Analysis of Data**. Now that the data is available and the statistical procedures have been specified, we need to apply them to actually compute the performance measures we seek. It is important here to compute not only point estimates for these performance measures but also error estimates, since the estimates are computed from a sample of data and therefore are subject to sampling variation.

12. **Implementation**. If all of the previous steps have been successful, we are ready to use the results to determine which decisions provide superior performance. Sometimes the model will be used for a one-time decision; other times, it is being developed so it will be available for other decisions, perhaps on a regular basis. When this is the case, the model must be implemented so it can be used routinely in the future.

13. **Final Documentation**. Often, the step of documenting the project, especially the model, is overlooked. However, this may be the most important part of the process. Even if further use of the model is not anticipated, it is important to document the model so that any future applications of the model can be effected without having to recreate it.

4 MODELING METHODOLOGY

Since the systems that are modeled are generally quite complex and dynamic, i.e., they change with time, special techniques have been developed to represent their structure and dynamics. In this section, we will discuss the characteristics of these models, classify them and review some of the methods for representing system models.

4.1 System Modeling

The *system state* is a collection of variables and possibly other information from the model that includes all information necessary for the model to operate over time. The major task in developing a simulation of a system is to come up with a model that captures the behavior of the system. Thus, the modeler must first select a representation for the system state. In some simple models, the system state could be a collection of variables. For example, in a simple queueing system, variables representing the number of customers in the queue and the status of the servers (idle or busy) would suffice. More realistic models, however, need a much richer representation for the system state in order to capture the complexity of the model. Most modeling approaches use the "entity-attribute-set" paradigm. In this view, a system is composed of *entities*. For example, in a hospital, the entities might be patients, doctors, nurses, operating rooms, X-ray equipment, etc. Entities have *attributes* which are items of information about the entity. A patient may have the attributes age, sex and type of illness. A doctor might have the attributes specialty and shift worked. Attributes are used to determine how the entities interact when the system model operates. A *set* (or queue) is a collection of entities, usually entities waiting for some resources so that their progress through the system can continue. Normally, sets belong to the system as a whole. For example, a hospital may have a set of patients waiting for surgery, a set of patients waiting for treatment in the emergency room and a set of patients occupying rooms. The system state then includes all entities present, the values of all of their attributes, their set memberships and any other system variables that may be defined.

4.1.1 Continuous, Discrete and Combined Systems

The modeler has two choices about how time can be represented - as a continuous variable or as a discrete variable. Some modeling approaches treat time as a continuous variable and express system changes in terms of a set of differential equations involving the system state variables. In this case, the simulation program numerically integrates the differential equations to compute the sample path of the system state. See chapter 15 of Pritsker (1986) for further discussion of continuous models.

For many systems the system state changes only at discrete points in time when an *event* occurs to cause a change. For example, in a hospital, the system state changes when a new patient arrives. The number of patients waiting for admission increases by 1, the newly arrived patient may join a queue and other actions occur to accomodate the patient. Other events that cause system changes are, for example, the discharge of a patient, the placing of a patient in a room, the beginning of surgery for a patient, the end of surgery, etc. Between any two events, the system state remains constant even though people are moving and services are being rendered. Thus, to represent the changes in the system, we only need to describe the actions that occur in each event. This type of model is called a *discrete-event system model.*

It is possible to have a combined discrete-event and continuous system model in which the values of some variables are controlled by differential equations and for others the values are changed at the moments that events occur. For example, in a steel mill, an event occurs to cause the steel to begin heating, but the temperature of the steel is determined by a set of differential equations that depend upon the amount of power applied, the starting temperature and some random variables. The event involving pouring the steel cannot occur until the steel exceeds a specified temperature. Thus, the variables that change continuously over time and those that change only in events are interdependent.

In practice, even variables that change continuously are evaluated only at discrete, usually equally-spaced, points in time since digital computers cannot represent continous functions numerically. Thus, we can consider even the continuous-time approach to be a special case of the discrete-event methodology, with the events denoting an updating of the values of the continuous variables.

4.1.2 Discrete-Event System Modeling

The dominant simulation modeling methodology is discrete-event system modeling. There are three basic ways this methodology is implemented in practice: the *event scheduling* approach, the *activity scanning* approach and the *process interaction* approach. While each of these methods has its own special way of looking at system dynamics, every discrete-event system can be modeled using any of the three methods.

The event scheduling approach represents model changes by identifying all types of events that may change the system state. For each type of event an event routine implements the actions to bring about the system changes. Events occur instantaneously. For example, in a hospital model, the events "arrival of a patient", "beginning of surgery", "end of surgery" and "discharge of patient" might be identified as the types of events that can occur. The modeler's job is to identify exactly what changes to the system occur in each type of event and provide the logic to effect those changes.

An *activity* is a period of time between two events, usually involving some action on an entity. For example, in a hospital, "surgery" is an activity which starts with the "beginning of surgery" event and ends with the "end of surgery" event. The time between the two events is the duration of the activity "surgery". Thus, for the activity scanning approach, our attention is on the activities that occur over time, rather than the events that start and end these activities. Some people find this a more natural way to think about system operations.

A *process* is a sequence of events and/or activities that are logically connected. For example, we can think of the process by which the surgery patient receives treatment in a hospital. First, the patient arrives to the hospital, then waits for surgery to begin. Next comes the surgery activity, followed by a recuperation period and ending with the discharge event. All events and activities relate to the same patient, but the patient processes for individual patients interact because patients are waiting for the same resources (admission clerks, doctors, operating rooms, etc.). Many modelers find this the most natural point of view for modeling, and most simulation languages adopt this approach.

Models can be expressed verbally or diagrammatically. Diagrammatic descriptions attempt to provide the same information as verbal descriptions but in a visual form. Event graphs (Schruben 1995) show the relationships among events using the event scheduling approach; GPSS flow diagrams are a visual representation of the process interaction approach as implemented by the GPSS simulation language. Visual model descriptions use the brain's visual information processing capabilities, which are older and more highly developed than our verbal information processing capabilities, to quickly describe the model.

5 SIMULATION SOFTWARE

Simulation software must be able to represent the system state and its changes over time. All discrete event simulation software packages have several characteristics in common. First, they are all able to represent entities, attributes and sets. All can create and destroy temporary entities, store them in one or more queues (sets), change the values of attributes, and represent resources used by the entities.

The dynamics of discrete-event simulations are implemented by creating a list of scheduled events, i.e., events that are known to occur at some time in the future. Even though a time may be random, once the value has been sampled from its distribution, it is known. The execution of the simulation is controlled by a *timing routine*, which selects the next event to occur and executes the appropriate event routine, activity actions or process actions for that event.

The following categories of simulation software differ in the interface with the user, the graphical capabilities and the ease of use. The choice of software, therefore, is primarily a matter of taste.

5.1 General Purpose Languages and Simulation Libraries

Perhaps most of the simulation programs that have ever been written were implemented using a general purpose language such as FORTRAN, Pascal, C, Algol, etc, augmented by a library of routines to implement the list processing, event list and random variate generation capabilities. Law and Kelton (1991) give an elementary collection of routines, which they called SIMLIB, for this purpose. Seila (1988) has developed a more extensive package called SIMTOOLS, that implements the event scheduling approach and the process interaction approach, for use with Pascal. Other packages have been developed for C, Pascal, Modula-2 and C++. This approach is very natural when used with modern object oriented languages such as C++ because the language can be very easily extended to include the simulation capabilities. See the other references in the bibliography for other packages of this sort that are available. This approach has the advantages that it is relatively inexpensive and the programmer does not have to learn a new language or new software system, but it often does not include the advanced graphical capabilities that are part of the other software packages.

5.2 Simulation Programming Languages

Starting in the 1960's, a generation of simulation programming languages developed which include capabilities for discrete-event simulation. One of the earliest of these was SIMULA, which was, and is, an advanced object-oriented language implementing the process interaction approach. Indeed, this language was very influential in the developement of modern object-oriented languages such as C++. Other early languages were GPSS and SIMSCRIPT. In fact, these languages developed contemporaneously with general-purpose languages and have all of the capabilities of general-purpose languages. These languages have evolved into modern versions and are widely used today. Tutorials on these and other simulation programming languages are given in the Software/Modelware Tracks. Simulation programming languages have the advantage that they allow the programmer to more naturally express the simulation actions in the language, rather than calling subprocedures to do the simulation actions. Some also include sophisticated graphical capabilities.

5.3 Interactive Simulation Programming Systems

Interactive simulation programming systems are an attempt to automate the process of translating a graphical model description into a simulation program. Examples of such systems are SIGMA (Schruben 1995) and CAPS/ECSL (Clementson 1991). In each of these systems, the analyst develops a graphical model representation, and the software converts this representation into a simulation program using a general-purpose language. The analyst has the opportunity to further edit the program before running it. Thus, interactive simulation programming systems are really a means to automate the development of a simulation program using a general purpose language. We should note, however, that the modeler expresses the simulation model *graphically*, so from the model implementation perspective, we start with a graphical model representation and end with a running simulation, and do not need to be concerned with the intermediate program if we do not wish to.

5.4 Visual Interactive Modeling Systems

The most recently introduced software for simulation utilize the graphical capabilities of modern computer workstations. These systems allow the model to be developed interactively and graphically, and then do model checking and execution directly from the graphical model or through an intermediate representation in a simulation language. ProModel and its derivatives (ServiceModel and MedModel), Arena, Witness and SIMFACTORY are examples of this category of simulation software. Some of these systems are called "data-driven simulators" because they are built on a generic network simulation model, and the user just provides data to specify the characteristics of each node in the network. It would be naive to leave the description at that, or to imply that all of these systems work in the same way. The important characteristic is that the user uses icons on screen to represent the system in a way that closely resembles the physical system. Thus, the model abstraction phase is largely obscured. Several of these packages are demonstrated in tutorials in the Software/Modelware tracks.

5.5 Simulation Program Verification

Regardless of the type of software that is used to implement the model, the analyst must make sure that the structure and logic of the implemented simulation corresponds to the intended model. For simulation programs this "debugging" process is complicated by the complexity of the system and the dynamic and uncertain nature of the operation of the program. Several techniques have been adopted to help the process. Random variates can be replaced with specific, known values so the response of the model can be predicted exactly and the program output can be compared to the predicted response. As with most complex software, simulation programs should be developed in modules that can be tested individually (the divide-and-conquer approach). Finally, animations of the systems are often useful to compare program operations to expected model behavior.

6 ANALYSIS METHODOLOGY

Probability and statistics play a dominant role in simulation methodology. In this section we will review the statistical methods that are very important in the practice of simulation.

6.1 Input Data Analysis

The term "input data analysis" refers to techniques to estimate parameters of the populations associated with random variables in the model and fit distributions to data collected from these populations in the real system. Most of the simulation texts in the bibliography contain complete descriptions of parameter estimation and distribution fitting methods. Often the modeler is only interested in finding, from among all possible candidates, the distribution that best fits a given set of data. Software is available to do this testing automatically. The Unifit II tutorial in the Software/Modelware track is one such package.

6.2 Random Variate Generation

The term *random variate* is used to mean an observation sampled on the computer from a specific distribution. Most simulation modelers do not need to be particularly concerned with the details of random variate generation techniques because all simulation software includes procedures to generate random variates from the commonly known distributions. All random variate generation methods begin with one or more independent, identically distributed uniform random variates between 0.0 and 1.0, and produce the desired random variate by making a transformation on this uniform random variate. Uniformly distributed random variates are generated by generating a uniformly distributed *integer* between 0 and m, where m is an upper limit related to the representation of integers on the computer. Therefore, the correctness and reliability of random variates depends fundamentally on the quality of the random number generator. Most popular simulation packages use random number generators that have been thoroughly tested and documented.

6.3 Output Data Analysis

The term "output data analysis" refers to techniques to estimate performance measures from data produced by the simulation as it runs. Unlike input data analysis, traditional statistical techniques are frequently not useful here. Whether traditional techniques can be used depends upon the design of the simulation runs. We could use a number of independent replications, each of short length, or we could use only one replication with a long run length. If all data used to estimate performance measures are statistically independent, then classical statistical methods can be used. However, this is frequently not the case.

If the system being simulated is stationary, the output processes will have an initial transient period that lasts until the processes settle into their stationary distributions. Part of output analysis involves determining how much of the data to truncate at the start before applying estimation methods. The stationary part of the output data generally is also autocorrelated. The effect of autocorrelation is usually to understate the size of confidence intervals for the mean, making the estimate appear to be much more precise than it really is. A number of techniques have been introduced to adjust for autocorrelation. One of the primary techniques used is the *batch means method*, in which one long output process is divided into several shorter segments whose sample means are computed and treated as a collection of mutually independent observations. This method has proved to be one of the most successful and robust techniques for computing reliable confidence intervals for the process mean from stationary output data. The tutorial on output analysis later in this volume reviews this method as well as other techniques for analyzing simulation data and provides additional references for this topic.

Frequently, the objective of data analysis is to compare two or more system designs to determine which provides better performance. Statistical methods for ranking and comparison are available to determine the reliability of the rankings produced by the simulations. System optimization methods seek to adjust one or more decision variables to optimize system performance. This is a new but promising area of research.

6.4 Validation

A model can be validated only if the real system exists. Techniques for validation are primarily statistical in nature and involve comparing the distributions of data from both the real system and the simulation to assure that they do not differ substantially. Two-sample tests for comparing distributions are available for this. Means, variances and other moments can also be compared and tested statistically. An informal technique for validation that has been used is to produce a report from the simulation that is identical in form to a report produced for the real system. The new report is then presented to someone who manages the real system and the manager is asked to see if she can detect whether it is from the real system or the simulation. Presumably, if they cannot detect that the report is from the simulation, the simulation is valid.

7 IMPLEMENTATION

The ultimate reward from developing a simulation model is to gain information that can be used to improve decision making with the system. In practice, the process of developing a simulation model is costly. Most models are a team effort and take months or even a year or more to develop. Costs are usually measured in tens of thousands of dollars. So, most organizations want the results of such an investment to be preserved. Some simulations are developed to be used on an on-going basis for system management and control by managers who were not necessarily involved with the original model development. In this case, documentation involves creating user's manuals and training managers to use the model. In all cases, documentation of the model makes sure the benefits of the model-building investment are not lost after the initial model use.

ACKNOWLEDGMENTS

Many people and many books and papers on the subject of simulation have influenced this paper. I would especially like to thank Professors George Fishman, Vlatko Čerić, Mike Pidd, David Goldsman and Christos Alexopoulos for many helpful discussions.

REFERENCES

Clementson, A. T. 1991. *The ECSL Plus System Manual.* A. T. Clementson, The Chestnuts, Princes Road, Windermere, Cumbria, UK.

Fishman, G. S. 1978. *Principles of Discrete Event Simulation.* New York: John Wiley and Sons.

Gogg, T. J., and J. R. A. Mott. 1993. Introduction to Simulation. In *Proceedings of the 1993 Winter Simulation Conference,* ed. G. W. Evans, M. Mollaghasemi, E. C. Russell, and W. E. Biles, 9–17. Institute of Electrical and Electronics Engineers, San Francisco, California.

Law, A. M., and W. D. Kelton. 1991. *Simulation Modeling and Analysis,* Second Edition. New York: McGraw-Hill.

Pagden, C. D., R. E. Shannon, and R. P. Sadowski. 1990. *Introduction to Simulation Using SIMAN.* New York: McGraw-Hill.

Pidd, M. 1994. An Introduction to Simulation. In *Proceedings of the 1994 Winter Simulation Conference,* ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila, 7–14. Institute of Electrical and Electronics Engineers, San Francisco, California.

Pritsker, A. A. B. 1986. *An Introduction to Simulation and SLAM II.* New York: John Wiley and Sons.

Schruben, L. W. 1995. *Graphical Simulation Modeling and Analysis Using SIGMA for Windows.* Danvers, MA: Boyd & Fraser.

Seila, A. F. 1988. SIMTOOLS: A Software Toolkit for Discrete Event Simulation in Pascal. *Simulation* 53.

Shannon, R. E. 1992. Introduction to Simulation. In *Proceedings of the 1992 Winter Simulation Conference,* ed. J. J. Swain, D. Goldsman, R. C. Crain and J. R. Wilson, 65–73. Institute of Electrical and Electronics Engineers, San Francisco, California.

Thesen, A. and L. E. Travis. 1991. Introduction to Simulation. In *Proceedings of the 1991 Winter Simulation Conference,* ed. B. L. Nelson, W. D. Kelton, and G. M. Clark. Institute of Electrical and Electronics Engineers, San Francisco, California.

AUTHOR BIOGRAPHY

Andrew F. Seila is associate professor of management science in the Terry College of Business at the University of Georgia, Athens, Georgia. He received a B.S. in Physics and Ph.D. in Operations Research and Systems Analysis, both from the University of North Carolina at Chapel Hill. Prior to joining the faculty of the University of Georgia, he was a Member Technical Staff at Bell Laboratories in Holmdel, New Jersey. His research interests include all areas of simulation modeling and analysis, especially output analysis. He has been actively involved in the Winter Simulation Conference since 1977, and served as Program Chair for the 1994 conference.