PART ONE

Introduction to Discrete-Event System Simulation

1

Introduction to Simulation

A *simulation* is the imitation of the operation of a real-world process or system over time. Whether done by hand or on a computer, simulation involves the generation of an artificial history of a system, and the observation of that artificial history to draw inferences concerning the operating characteristics of the real system.

The behavior of a system as it evolves over time is studied by developing a simulation *model*. This model usually takes the form of a set of assumptions concerning the operation of the system. These assumptions are expressed in mathematical, logical, and symbolic relationships between the *entities*, or objects of interest, of the system. Once developed and validated, a model can be used to investigate a wide variety of "what-if" questions about the real-world system. Potential changes to the system can first be simulated in order to predict their impact on system performance. Simulation can also be used to study systems in the design stage, before such systems are built. Thus, simulation modeling can be used both as an analysis tool for predicting the effect of changes to existing systems, and as a design tool to predict the performance of new systems under varying sets of circumstances.

In some instances, a model can be developed which is simple enough to be "solved" by mathematical methods. Such solutions may be found by the use of differential calculus, probability theory, algebraic methods, or other mathematical techniques. The solution usually consists of one or more numerical parameters which are called *measures of performance* of the system. However, many real-world systems are so complex that models of these systems are virtually impossible to solve mathematically. In these instances, numerical, computer-based simulation can be used to imitate the behavior of the system

over time. From the simulation, data are collected as if a real system were being observed. This simulation-generated data is used to estimate the measures of performance of the system.

This book provides an introductory treatment of the concepts and methods of one form of simulation modeling—discrete-event simulation modeling. The first chapter initially discusses when to use simulation, its advantages and disadvantages, and actual areas of application. Then the concepts of system and model are explored. Finally, an outline is given of the steps in building and using a simulation model of a system.

1.1 When Is Simulation the Appropriate Tool?

The availability of special-purpose simulation languages, massive computing capabilities at a decreasing cost per operation, and advances in simulation methodologies have made simulation one of the most widely used and accepted tools in operations research and systems analysis. Circumstances under which simulation is the appropriate tool to use have been discussed by many authors, from Naylor et al. [1966] to Banks et al. [1996]. Simulation can be used for the following purposes:

1. Simulation enables the study of, and experimentation with, the internal interactions of a complex system, or of a subsystem within a complex system.
2. Informational, organizational, and environmental changes can be simulated, and the effect of these alterations on the model's behavior can be observed.
3. The knowledge gained in designing a simulation model may be of great value toward suggesting improvement in the system under investigation.
4. By changing simulation inputs and observing the resulting outputs, valuable insight may be obtained into which variables are most important and how variables interact.
5. Simulation can be used as a pedagogical device to reinforce analytic solution methodologies.
6. Simulation can be used to experiment with new designs or policies prior to implementation, so as to prepare for what may happen.
7. Simulation can be used to verify analytic solutions.
8. By simulating different capabilities for a machine, requirements can be determined.
9. Simulation models designed for training allow learning without the cost and disruption of on-the-job learning.

10. Animation shows a system in simulated operation so that the plan can be visualized.

11. The modern system (factory, wafer fabrication plant, service organization, etc.) is so complex that the interactions can be treated only through simulation.

1.2 When Simulation Is Not Appropriate

This section is based on an article by Banks and Gibson [1997], who gave ten rules for determining when simulation is not appropriate. The first rule indicates that simulation should not be used when the problem can be solved using common sense. An example is given of an automobile-tag facility serving customers who arrive randomly at an average rate of 100/hour and are served at a mean rate of 12/hour. To determine the minimum number of servers needed, simulation is not necessary. Just compute 100/12 = 8.33, indicating that nine or more servers are needed.

The second rule says that simulation should not be used if the problem can be solved analytically. For example, under certain conditions, the average waiting time in the example above can be determined from curves that were developed by Hillier and Lieberman [1995].

The next rule says that simulation should not be used if it is easier to perform direct experiments. An example of a fast-food drive-in restaurant is given, where it was less expensive to have a person use a hand-held terminal and voice communication to determine the effect of adding another order station on customer waiting time.

The fourth rule says not to use simulation, if the costs exceed the savings. There are many steps in completing a simulation as discussed in Section 1.11, and these must be done thoroughly. If a simulation study costs $20,000 and the savings might be $10,000, simulation would not be appropriate.

Rules five and six indicate that simulation should not be performed if the resources or time are not available. If the simulation is estimated to cost $20,000 and only $10,000 is available, the suggestion is not to venture into a simulation study. Similarly, if a decision in needed is two weeks and a simulation will take a month, the simulation study is not advised.

Simulation takes data, sometimes lots of data. If no data is available, not even estimates, simulation is not advised.

The next rule concerns the ability to verify and validate the model. If there is not enough time or the personnel are not available, simulation is not appropriate.

If managers have unreasonable expectations—say, too much too soon—or the power of simulation is overestimated, simulation may not be appropriate.

Last, if system behavior is too complex or can't be defined, simulation is not appropriate. Human behavior is sometimes extremely complex to model.

1.3 Advantages and Disadvantages of Simulation

Simulation is intuitively appealing to a client because it mimics what happens in a real system or what is perceived for a system that is in the design stage. The output data from a simulation should directly correspond to the outputs that could be recorded from the real system. Additionally, it is possible to develop a simulation model of a system without dubious assumptions (such as the same statistical distribution for every random variable) of mathematically solvable models. For these, and other reasons, simulation is frequently the technique of choice in problem solving.

In contrast to optimization models, simulation models are "run" rather than solved. Given a particular set of input and model characteristics, the model is run and the simulated behavior is observed. This process of changing inputs and model characteristics results in a set of scenarios that are evaluated. A good solution, either in the analysis of an existing system or the design of a new system, is then recommended for implementation.

Simulation has many advantages, and even some disadvantages. These are listed by Pegden, Shannon, and Sadowski [1995]. The advantages are:

1. New policies, operating procedures, decision rules, information flows, organizational procedures, and so on can be explored without disrupting ongoing operations of the real system.
2. New hardware designs, physical layouts, transportation systems, and so on, can be tested without committing resources for their acquisition.
3. Hypotheses about how or why certain phenomena occur can be tested for feasibility.
4. Time can be compressed or expanded allowing for a speedup or slowdown of the phenomena under investigation.
5. Insight can be obtained about the interaction of variables.
6. Insight can be obtained about the importance of variables to the performance of the system.
7. Bottleneck analysis can be performed indicating where work-in-process, information, materials, and so on are being excessively delayed.
8. A simulation study can help in understanding how the system operates rather than how individuals think the system operates.
9. "What-if" questions can be answered. This is particularly useful in the design of new systems.

The disadvantages are:

1. Model building requires special training. It is an art that is learned over time and through experience. Furthermore, if two models are constructed by two competent individuals, they may have similarities, but it is highly unlikely that they will be the same.

2. Simulation results may be difficult to interpret. Since most simulation outputs are essentially random variables (they are usually based on random inputs), it may be hard to determine whether an observation is a result of system interrelationships or randomness.
3. Simulation modeling and analysis can be time consuming and expensive. Skimping on resources for modeling and analysis may result in a simulation model or analysis that is not sufficient for the task.
4. Simulation is used in some cases when an analytical solution is possible, or even preferable, as discussed in Section 1.2. This might be particularly true in the simulation of some waiting lines where closed-form queueing models are available.

In defense of simulation, these four disadvantages, respectively, can be offset as follows:

1. Vendors of simulation software have been actively developing packages that contain models that need only input data for their operation. Such models have the generic tag "simulators" or "templates."
2. Many simulation software vendors have developed output analysis capabilities within their packages for performing very thorough analysis.
3. Simulation can be performed faster today than yesterday, and even faster tomorrow. This is attributable to the advances in hardware that permit rapid running of scenarios. It is also attributable to the advances in many simulation packages. For example, some simulation software contains constructs for modeling material handling using transporters such as fork lift trucks, conveyors, automated guided vehicles, and others.
4. Closed-form models are not able to analyze most of the complex systems that are encountered in practice. In nearly eight years of consulting practice by one of the authors, not one problem was encountered that could have been solved by a closed-form solution.

1.4 Areas of Application

The applications of simulation are vast. The Winter Simulation Conference (WSC) is an excellent way to learn more about the latest in simulation applications and theory. There are also numerous tutorials at both the beginning and advanced levels. WSC is sponsored by six technical societies and the National Institute of Standards and Technology (NIST). The technical societies are American Statistical Association (ASA), Association for Computing Machinery/Special Interest Group on Simulation (ACM/SIGSIM), Institute of Electrical and Electronics Engineers: Computer Society (IEEE/CS), Institute of Electrical and Electronics Engineers: Systems, Man and Cybernetics Society (IEEE/SMCS), Institute of Industrial Engineers (IIE), Institute for Operations Research and the Management Sciences: College on Simulation (INFORMS/CS), and The Society for Computer Simulation (SCS). Note that

IEEE is represented by two bodies. Information about the upcoming WSC can be obtained from

www.wintersim.org

Some of the areas of application at a recent WSC, with the subject matter within those areas, is listed below:

Manufacturing Applications

Analysis of electronics assembly operations
Design and evaluation of a selective assembly station for high-precision scroll compressor shells
Comparison of dispatching rules for semiconductor manufacturing using large-facility models
Evaluation of cluster tool throughput for thin-film head production
Determining optimal lot size for a semiconductor back-end factory
Optimization of cycle time and utilization in semiconductor test manufacturing
Analysis of storage and retrieval strategies in a warehouse
Investigation of dynamics in a service-oriented supply chain
Model for an Army chemical munitions disposal facility

Semiconductor Manufacturing

Comparison of dispatching rules using large-facility models
The corrupting influence of variability
A new lot-release rule for wafer fabs
Assessment of potential gains in productivity due to proactive reticle management
Comparison of a 200-mm and 300-mm X-ray lithography cell
Capacity planning with time constraints between operations
300-mm logistic system risk reduction

Construction Engineering

Construction of a dam embankment
Trenchless renewal of underground urban infrastructures
Activity scheduling in a dynamic, multiproject setting
Investigation of the structural steel erection process
Special-purpose template for utility tunnel construction

Military Applications

Modeling leadership effects and recruit type in an Army recruiting station
Design and test of an intelligent controller for autonomous underwater vehicles
Modeling military requirements for nonwarfighting operations
Multitrajectory performance for varying scenario sizes
Using adaptive agents in U.S. Air Force pilot retention

Logistics, Transportation, and Distribution Applications

Evaluating the potential benefits of a rail-traffic planning algorithm
Evaluating strategies to improve railroad performance
Parametric modeling in rail-capacity planning
Analysis of passenger flows in an airport terminal
Proactive flight-schedule evaluation
Logistics issues in autonomous food production systems for extended-duration space exploration
Sizing industrial rail-car fleets
Product distribution in the newspaper industry
Design of a toll plaza
Choosing between rental-car locations
Quick-response replenishment

Business Process Simulation

Impact of connection bank redesign on airport gate assignment
Product development program planning
Reconciliation of business and systems modeling
Personnel forecasting and strategic workforce planning

Human Systems

Modeling human performance in complex systems
Studying the human element in air traffic control

1.5 Systems and System Environment

To model a system, it is necessary to understand the concept of a system and the system boundary. A *system* is defined as a group of objects that are joined together in some regular interaction or interdependence toward the accomplishment of some purpose. An example is a production system manufacturing automobiles. The machines, component parts, and workers operate jointly along an assembly line to produce a high-quality vehicle.

A system is often affected by changes occurring outside the system. Such changes are said to occur in the *system environment* [Gordon, 1978]. In modeling systems, it is necessary to decide on the *boundary* between the system and its environment. This decision may depend on the purpose of the study.

> In the case of the factory system, for example, the factors controlling the arrival of orders may be considered to be outside the influence of the factory and therefore part of the environment. However, if the effect of supply on demand is to be considered, there will be a relationship between factory output and arrival of orders, and this relationship must be considered an activity of the system. Similarly, in the case of a bank system, there may be a limit on the maximum interest rate that can be paid. For the study of a single bank, this would be regarded as a constraint imposed by the environment. In a study of the effects of monetary laws on the banking industry, however, the setting of the limit would be an activity of the system. [Gordon, 1978]

1.6 Components of a System

In order to understand and analyze a system, a number of terms need to be defined. An *entity* is an object of interest in the system. An *attribute* is a property of an entity. An *activity* represents a time period of specified length. If a bank is being studied, customers might be one of the entities, the balance in their checking accounts might be an attribute, and making deposits might be an activity.

The collection of entities that compose a system for one study might be only a subset of the overall system for another study [Law and Kelton, 2000]. For example, if the bank mentioned above is being studied to determine the number of tellers needed to provide for paying and receiving, the system can be defined as that portion of the bank consisting of the regular tellers and the customers waiting in line. If the purpose of the study is expanded to determine the number of special tellers needed (to prepare cashier's checks, to sell traveler's checks, etc.), the definition of the system must be expanded.

The *state* of a system is defined to be that collection of variables necessary to describe the system at any time, relative to the objectives of the study. In the study of a bank, possible state variables are the number of busy tellers, the number of customers waiting in line or being served, and the arrival time of the next customer. An *event* is defined as an instantaneous occurrence that may change the state of the system. The term *endogenous* is used to describe activities and events occurring within a system, and the term *exogenous* is used to describe activities and events in the environment that affect the system. In the bank study, the arrival of a customer is an exogenous event, and the completion of service of a customer is an endogenous event.

Table 1.1 lists examples of entities, attributes, activities, events, and state variables for several systems. Only a partial listing of the system components is shown. A complete list cannot be developed unless the purpose of the study is known. Depending on the purpose, various aspects of the system will be of interest, and then the listing of components can be completed.

Table 1.1 Examples of Systems and Components

System	*Entities*	*Attributes*	*Activities*	*Events*	*State Variables*
Banking	Customers	Checking account balance	Making deposits	Arrival; departure	Number of busy tellers; number of customers waiting
Rapid rail	Riders	Origination; destination	Traveling	Arrival at station; arrival at destination	Number of riders waiting at each station; number of riders in transit
Production	Machines	Speed; capacity; breakdown rate	Welding; stamping	Breakdown	Status of machines (busy, idle, or down)
Communications	Messages	Length; destination	Transmitting	Arrival at destination	Number waiting to be transmitted
Inventory	Warehouse	Capacity	Withdrawing	Demand	Levels of inventory; backlogged demands

1.7 Discrete and Continuous Systems

Systems can be categorized as discrete or continuous. "Few systems in practice are wholly discrete or continuous, but since one type of change predominates for most systems, it will usually be possible to classify a system as being either discrete or continuous" [Law and Kelton, 2000]. A *discrete system* is one in which the state variable(s) change only at a discrete set of points in time. The bank is an example of a discrete system since the state variable, the number of customers in the bank, changes only when a customer arrives or when the service provided a customer is completed. Figure 1.1 shows how the number of customers changes only at discrete points in time.

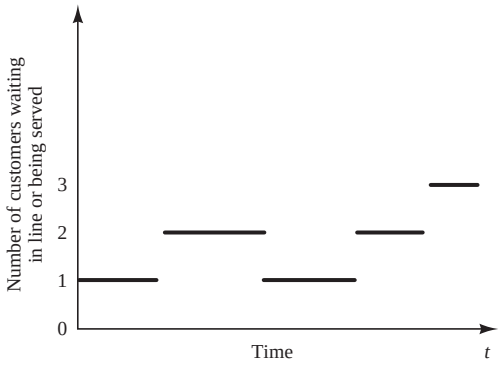


Figure 1.1 Discrete-system state variable.

A *continuous system* is one in which the state variable(s) change continuously over time. An example is the head of water behind a dam. During and for some time after a rain storm, water flows into the lake behind the dam. Water is drawn from the dam for flood control and to make electricity. Evaporation also decreases the water level. Figure 1.2 shows how the state variable, head of water behind the dam, changes for this continuous system.

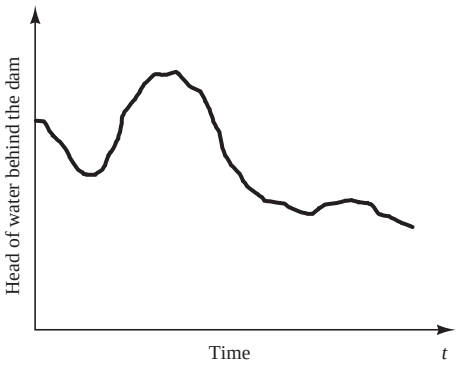


Figure 1.2 Continuous-system state variable.

1.8 Model of a System

Sometimes it is of interest to study a system to understand the relationships between its components or to predict how the system will operate under a new policy. Sometimes it is possible to experiment with the system itself, but, not always. A new system may not yet exist; it may be only in hypothetical form or at the design stage. Even if the system exists, it may be impractical to experiment with it. For example, it may not be wise or possible to double the unemployment rate to determine the effect of employment on inflation. In the case of a bank, reducing the numbers of tellers to study the effect on the length of waiting lines may infuriate the customers so greatly that they move their accounts to a competitor. Consequently, studies of systems are often accomplished with a model of a system.

We had a consulting job for the simulation of a redesigned port in western Australia. At $200 million for a loading/unloading berth, it's not advisable to invest that amount only to find that the berth is inadequate for the task.

A *model* is defined as a representation of a system for the purpose of studying the system. For most studies, it is necessary to consider only those aspects of the system that affect the problem under investigation. These aspects are represented in a model of the system, and the model, by definition, is a simplification of the system. On the other hand, the model should be sufficiently detailed to permit valid conclusions to be drawn about the real system. Different models of the same system may be required as the purpose of investigation changes.

Just as the components of a system were entities, attributes, and activities, models are represented similarly. However, the model contains only those components that are relevant to the study. The components of a model are discussed more extensively in Chapter 3.

1.9 Types of Models

Models can be classified as being mathematical or physical. A mathematical model uses symbolic notation and mathematical equations to represent a system. A simulation model is a particular type of mathematical model of a system.

Simulation models may be further classified as being static or dynamic, deterministic or stochastic, and discrete or continuous. A *static* simulation model, sometimes called a Monte Carlo simulation, represents a system at a particular point in time. *Dynamic* simulation models represent systems as they change over time. The simulation of a bank from 9:00 A.M. to 4:00 P.M. is an example of a dynamic simulation.

Simulation models that contain no random variables are classified as *deterministic*. Deterministic models have a known set of inputs which will result in a unique set of outputs. Deterministic arrivals would occur at a dentist's office

if all patients arrived at the scheduled appointment time. A *stochastic* simulation model has one or more random variables as inputs. Random inputs lead to random outputs. Since the outputs are random, they can be considered only as estimates of the true characteristics of a model. The simulation of a bank would usually involve random interarrival times and random service times. Thus, in a stochastic simulation, the output measures—the average number of people waiting, the average waiting time of a customer—must be treated as statistical estimates of the true characteristics of the system.

Discrete and continuous systems were defined in Section 1.6. Discrete and continuous models are defined in an analogous manner. However, a discrete simulation model is not always used to model a discrete system, nor is a continuous simulation model always used to model a continuous system. Tanks and pipes are modeled discretely by some software vendors, even though we know that fluid flow is continuous. In addition, simulation models may be mixed, both discrete and continuous. The choice of whether to use a discrete or continuous (or both discrete and continuous) simulation model is a function of the characteristics of the system and the objective of the study. Thus, a communication channel could be modeled discretely if the characteristics and movement of each message were deemed important. Conversely, if the flow of messages in aggregate over the channel were of importance, modeling the system using continuous simulation could be more appropriate. The models considered in this text are discrete, dynamic, and stochastic.

1.10 Discrete-Event System Simulation

This book is about discrete-event system simulation—the modeling of systems in which the state variable changes only at a discrete set of points in time. The simulation models are analyzed by numerical rather than by analytical methods. *Analytical* methods employ the deductive reasoning of mathematics to "solve" the model. For example, differential calculus can be used to determine the minimum-cost policy for some inventory models. *Numerical* methods employ computational procedures to "solve" mathematical models. In the case of simulation models, which employ numerical methods, models are "run" rather than solved; that is, an artificial history of the system is generated based on the model assumptions, and observations are collected to be analyzed and to estimate the true system performance measures. Since real-world simulation models are rather large, and since the amount of data stored and manipulated is so vast, the runs are usually conducted with the aid of a computer. However, much insight can be obtained by simulating small models manually.

In summary, this book is about discrete-event system simulation in which the models of interest are analyzed numerically, usually with the aid of a computer.

1.11 Steps in a Simulation Study

Figure 1.3 shows a set of steps to guide a model builder in a thorough and sound simulation study. Similar figures and discussion of steps can be found in other sources [Shannon, 1975; Gordon, 1978; Law and Kelton, 2000]. The number beside each symbol in Figure 1.3 refers to the more detailed discussion in the text. The steps in a simulation study are as follows:

Problem formulation. Every study should begin with a statement of the problem. If the statement is provided by the policy makers, or those that have the problem, the analyst must ensure that the problem being described is clearly understood. If a problem statement is being developed by the analyst, it is important that the policy makers understand and agree with the formulation. Although not shown in Figure 1.3, there are occasions where the problem must be reformulated as the study progresses. In many instances, policy makers and analysts are aware that there is a problem long before the nature of the problem is known.

Setting of objectives and overall project plan. The objectives indicate the questions to be answered by simulation. At this point a determination should be made concerning whether simulation is the appropriate methodology for the problem as formulated and objectives as stated. Assuming it is decided that simulation is appropriate, the overall project plan should include a statement of the alternative systems to be considered, and a method for evaluating the effectiveness of these alternatives. It should also include the plans for the study in terms of the number of people involved, the cost of the study, and the number of days required to accomplish each phase of the work with the anticipated results at the end of each stage.

Model conceptualization. The construction of a model of a system is probably as much art as science. Pritsker [1998] provides a lengthy discussion of this step. "Although it is not possible to provide a set of instructions that will lead to building successful and appropriate models in every instance, there are some general guidelines that can be followed" [Morris, 1967]. The art of modeling is enhanced by an ability to abstract the essential features of a problem, to select and modify basic assumptions that characterize the system, and then to enrich and elaborate the model until a useful approximation results. Thus, it is best to start with a simple model and build toward greater complexity. However, the model complexity need not exceed that required to accomplish the purposes for which the model is intended. Violation of this principle will only add to model-building and computer expenses. It is not necessary to have a one-to-one mapping between the model and the real system. Only the essence of the real system is needed.

It is advisable to involve the model user in model conceptualization. This will both enhance the quality of the resulting model and increase the confidence of the model user in the application of the model. (Chapter 2 describes a

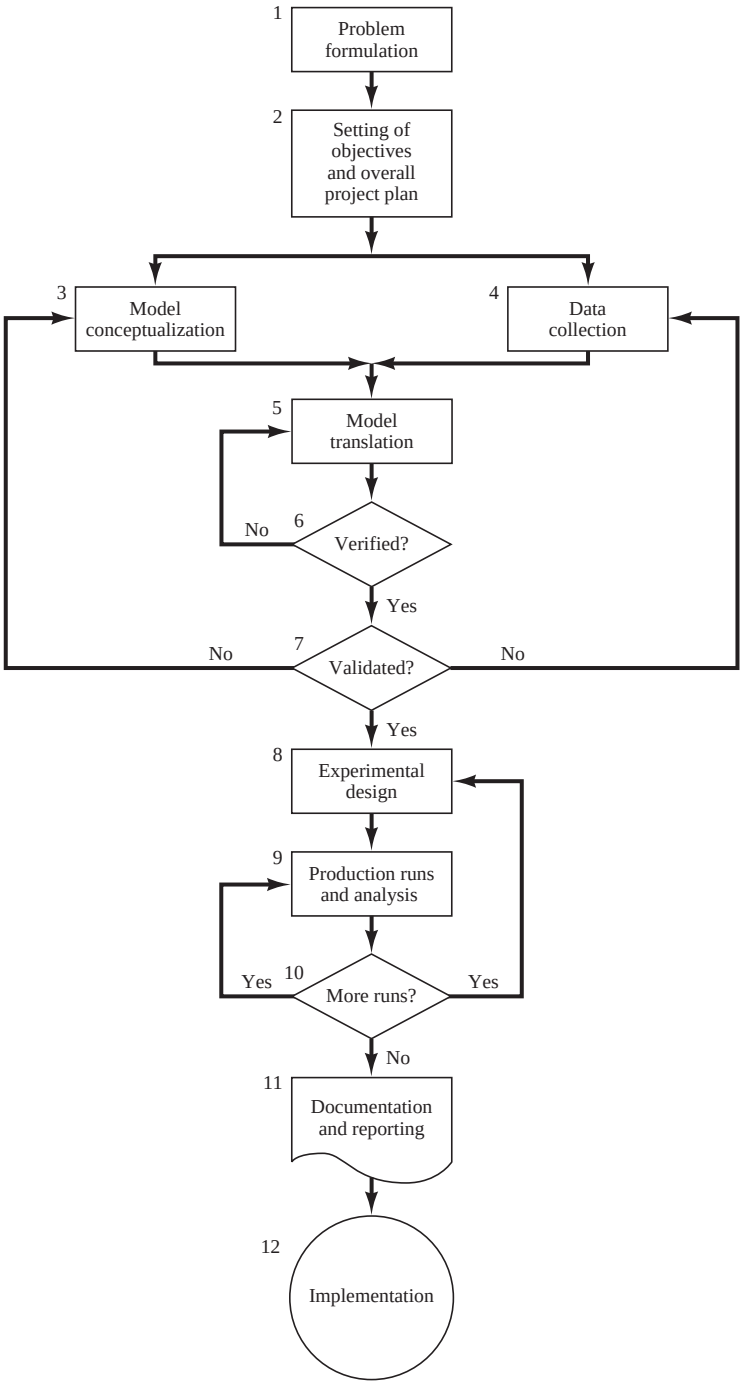


Figure 1.3 Steps in a simulation study.

number of simulation models. Chapter 6 describes queueing models that can be solved analytically. However, only experience with real systems—versus textbook problems—can "teach" the art of model building.)

Data collection. There is a constant interplay between the construction of the model and the collection of the needed input data [Shannon, 1975]. As the complexity of the model changes, the required data elements may also change. Also, since data collection takes such a large portion of the total time required to perform a simulation, it is necessary to begin it as early as possible, usually together with the early stages of model building.

The objectives of the study dictate, in a large way, the kind of data to be collected. In the study of a bank, for example, if the desire is to learn about the length of waiting lines as the number of tellers change, the types of data needed would be the distributions of interarrival times (at different times of the day), the service-time distributions for the tellers, and historic distributions on the lengths of waiting lines under varying conditions. These last data will be used to validate the simulation model. (Chapter 9 discusses data collection and data analysis; Chapter 5 discusses statistical distributions which occur frequently in simulation modeling. See also an excellent discussion by Vincent [1998].)

Model translation. Since most real-world systems result in models that require a great deal of information storage and computation, the model must be entered into a computer-recognizable format. We use the term "program," even though it is possible to accomplish the desired result in many instances with little or no actual coding. The modeler must decide whether to program the model in a simulation language such as GPSS/H© (discussed in Chapter 4) or to use special-purpose simulation software. For manufacturing and material handling, Chapter 4 discusses Arena®, AutoMod©, CSIM, Extend©, Micro Saint, ProModel®, Dened/Quest®, Taylor Eneterprise Dynamics (ED), and Witness©. Simulation languages are powerful and flexible. However, if the problem is amenable to solution with the simulation software, the model development time is greatly reduced. Furthermore, most of the simulation software packages have added features that enhance their flexibility, although the amount of flexibility varies greatly.

Verified? Verification pertains to the computer program prepared for the simulation model. Is the computer program performing properly? With complex models it is difficult, if not impossible, to translate a model successfully in its entirety without a good deal of debugging. If the input parameters and logical structure of the model are correctly represented in the computer, verification has been completed. For the most part, common sense is used in completing this step. (Chapter 10 discusses verification of simulation models, and Balci [1998] also discusses this topic extensively.)

Validated? Validation is the determination that a model is an accurate representation of the real system. Validation is usually achieved through the calibration of the model, an iterative process of comparing the model to actual system behavior and using the discrepancies between the two, and the insights

gained, to improve the model. This process is repeated until model accuracy is judged acceptable. In the example of a bank mentioned above, data were collected concerning the length of waiting lines under current conditions. Does the simulation model replicate this system measure? This is one means of validation. (Chapter 10 discusses the validation of simulation models, and Balci [1998] also discusses this topic extensively.)

Experimental design. The alternatives that are to be simulated must be determined. Often, the decision concerning which alternatives to simulate may be a function of runs that have been completed and analyzed. For each system design that is simulated, decisions need to be made concerning the length of the initialization period, the length of simulation runs, and the number of replications to be made of each run. (Chapter 11 and 12 discuss issues associated with the experimental design, and Kleijnen [1998] discusses this topic extensively.)

Production runs and analysis. Production runs, and their subsequent analysis, are used to estimate measures of performance for the system designs that are being simulated. [Chapters 11 and 12 discuss the analysis of simulation experiments, and Chapter 4 discusses software to aid in this step, including AutoStat (in AutoMod), OptQuest (in several simulation softwares), SimRunner (in ProModel) and the Arena Output Analyzer.]

More Runs? Based on the analysis of runs that have been completed, the analyst determines if additional runs are needed and what design those additional experiments should follow.

Documentation and reporting. There are two types of documentation: program and progress. Program documentation is necessary for numerous reasons. If the program is going to be used again by the same or different analysts, it may be necessary to understand how the program operates. This will build confidence in the program, so that model users and policy makers can make decisions based on the analysis. Also, if the program is to be modified by the same or a different analyst, this can be greatly facilitated by adequate documentation. One experience with an inadequately documented program is usually enough to convince an analyst of the necessity of this important step. Another reason for documenting a program is so that model users can change parameters at will in an effort to determine the relationships between input parameters and output measures of performance, or to determine the input parameters that "optimize" some output measure of performance.

Musselman [1998] discusses progress reports that provide the important, written history of a simulation project. Project reports give a chronology of work done and decisions made. This can prove to be of great value in keeping the project on course.

Musselman suggests frequent reports (monthly, at least) so that even those not involved in the day-to-day operation can keep abreast. The awareness of these others can usually enhance the successful completion of the project

by surfacing misunderstandings early, when the problem can be solved easily. Musselman also suggests maintaining a project log providing a comprehensive record of accomplishments, change requests, key decisions, and other items of importance.

On the reporting side, Musselman suggests frequent deliverables. These may or may not be the results of major accomplishments. His maxim is that "it is better to work with many intermediate milestones than with one absolute deadline." Possibilities prior to the final report include a model specification, prototype demonstrations, animations, training results, intermediate analyses, program documentation, progress reports, and presentations. He suggests that these deliverables should be timed judiciously over the life of the project.

The result of all the analysis should be reported clearly and concisely in a final report. This will enable the model users (now, the decision makers) to review the final formulation, the alternative systems that were addressed, the criterion by which the alternatives were compared, the results of the experiments, and the recommended solution to the problem. Furthermore, if decisions have to be justified at a higher level, the final report should provide a vehicle of certification for the model user/decision maker and add to the credibility of the model and the model-building process.

Implementation. The success of the implementation phase depends on how well the previous eleven steps have been performed. It is also contingent upon how thoroughly the analyst has involved the ultimate model user during the entire simulation process. If the model user has been thoroughly involved and understands the nature of the model and its outputs, the likelihood of a vigorous implementation is enhanced [Pritsker, 1995]. Conversely, if the model and its underlying assumptions have not been properly communicated, implementation will probably suffer, regardless of the simulation model's validity.

The simulation model-building process shown in Figure 1.3 can be broken down into four phases. The first phase, consisting of steps 1 (Problem Formulation) and 2 (Setting of Objective and Overall Design), is a period of discovery or orientation. The initial statement of the problem is usually quite "fuzzy," the initial objectives will usually have to be reset, and the original project plan will usually have to be fine-tuned. These recalibrations and clarifications may occur in this phase, or perhaps after or during another phase (i.e., the analyst may have to restart the process).

The second phase is related to model building and data collection and includes steps 3 (Model Conceptualization), 4 (Data Collection), 5 (Model Translation), 6 (Verification), and 7 (Validation). A continuing interplay is required among the steps. Exclusion of the model user during this phase can have dire implications at the point of implementation.

The third phase concerns running the model. It involves steps 8 (Experimental Design), 9 (Production Runs and Analysis), and 10 (Additional Runs). This phase must have a thoroughly conceived plan for experimenting with the simulation model. A discrete-event stochastic simulation is in fact a statisti-

cal experiment. The output variables are estimates that contain random error, and therefore a proper statistical analysis is required. Such a philosophy differs sharply from that of the analyst who makes a single run and draws an inference from that single data point.

The fourth phase, implementation, involves steps 11 (Documentation and Reporting) and 12 (Implementation). Successful implementation depends on continual involvement of the model user and the successful completion of every step in the process. Perhaps the most crucial point in the entire process is step 7 (Validation), because an invalid model is going to lead to erroneous results, which if implemented could be dangerous, costly, or both.

REFERENCES

Balci, Osman [1998], "Verification, Validation, and Testing," in *Handbook of Simulation*, ed. Jerry Banks, John Wiley, New York.

Banks, Jerry, and Randall R. Gibson [1997], "Don't Simulate When: 10 rules for determining when simulation is not appropriate," *IIE Solutions*, September.

Banks, Jerry, Mark Spearman, and Van Norman [1996], "Second Look at Simulation," *OR/MS Today*, Vol. 22, No. 4, August 1996.

Gordon, Geoffrey [1978], *System Simulation*, 2d ed., Prentice-Hall, Upper Saddle River, NJ.

Hillier, Frederick S., and Gerald J. Lieberman [1995], *Introduction to Operations Research*, 6th ed., McGraw-Hill, New York.

Kleijnen, Jack P. C. [1998], "Experimental Design for Sensitivity Analysis, Optimization, and Validation of Simulation Models," in *Handbook of Simulation*, ed. Jerry Banks, John Wiley, New York.

Law, Averill M., and W. David Kelton [2000], *Simulation Modeling and Analysis*, 3d ed., McGraw-Hill, New York.

Morris, W. T. [1967], "On the Art of Modeling," *Management Science*, Vol. 13, No. 12.

Musselman, Kenneth J. [1998], "Guidelines for Success," in *Handbook of Simulation*, ed. Jerry Banks, John Wiley, New York.

Naylor, T. H., J. L. Balintfy, D. S. Burdick, and K. Chu [1966], *Computer Simulation Techniques*, John Wiley, New York.

Pegden, C. D., R. E. Shannon, and R. P. Sadowski [1995], *Introduction to Simulation Using SIMAN*, 2d ed., McGraw-Hill, New York.

Pritsker, A. Alan B. [1995], *Introduction to Simulation and SLAM II*, 4th ed., John Wiley, New York.

Pritsker, A. Alan B. [1998], "Principles of Simulation Modeling," in *Handbook of Simulation*, ed. Jerry Banks, John Wiley, New York.

Shannon, Robert E. [1975], *Systems Simulation: The Art and Science*, Prentice-Hall, Upper Saddle River, NJ.

Vincent, Stephen [1998], "Input Data Analysis," in *Handbook of Simulation*, ed. Jerry Banks, John Wiley, New York.

EXERCISES

1 Name several entities, attributes, activities, events, and state variables for the following systems:

- **(a)** A small appliance repair shop
- **(b)** A cafeteria
- **(c)** A grocery store
- **(d)** A laundromat
- **(e)** A fast-food restaurant
- **(f)** A hospital emergency room
- **(g)** A taxicab company with 10 taxis
- **(h)** An automobile assembly line

2 Consider the simulation process shown in Figure 1.3.

- **(a)** Reduce the steps by at least two by combining similar activities. Give your rationale.
- **(b)** Increase the steps by at least two by separating current steps or enlarging on existing steps. Give your rationale.

3 A simulation of a major traffic intersection is to be conducted with the objective of improving the current traffic flow. Provide three iterations, in increasing order of complexity, of steps 1 and 2 in the simulation process of Figure 1.3.

4 In what ways and at what steps might a personal computer be used to support the simulation process of Figure 1.3?

5 A simulation is to be conducted of cooking a spaghetti dinner to determine what time a person should start in order to have the meal on the table by 7:00 P.M. Read a recipe for preparing a spaghetti dinner (or ask a friend or relative, etc., for the recipe). As best you can, trace what you understand to be needed in the data-collection phase of the simulation process of Figure 1.3 in order to perform a simulation in which the model includes each step in the recipe. What are the events, activities, and state variables in this system?

6 What events and activities are associated with the operation of your checkbook?

7 Read an article in the current *WSC Proceedings*, on the application of simulation related to your major area of study or interest, and prepare a report on how the author accomplishes the steps given in Figure 1.3. *WSC Proceedings* are available at

`www.informs-cs.org`

8 Get a copy of a recent *WSC Proceedings* and report on the different applications discussed in an area of interest to you. *WSC Proceedings* are available at

`www.informs-cs.org`

9 Get a copy of a recent *WSC Proceedings* and report on the most unusual application that you can find. *WSC Proceedings* are available at

`www.informs-cs.org`

10 Go to the Simulation Education website at

`www.pitt.edu/@wjyst/nsfteachsim.html`

and address the following:

(a) Use the links there to thoroughly answer the question, "What is simulation?"

(b) What kinds of careers are available in simulation?

(c) What are some recent applications of discrete-event simulation in the news?

(d) What are some simulation education organizations that are currently active?

11 Go to the Winter Simulation Conference website at

`www.wintersim.org`

and address the following:

(a) What advanced tutorials were offered at the previous WSC or are planned at the next WSC?

(b) Where and when will the next WSC be held?

(c) What is the history of WSC?

2

Simulation Examples

This chapter presents several examples of simulations that can be performed by devising a simulation table either manually or with a spreadsheet. The simulation table provides a systematic method for tracking system state over time. These examples provide insight into the methodology of discrete system simulation and the descriptive statistics used for predicting system performance.

The simulations in this chapter entail three steps:

1. Determine the characteristics of each of the inputs to the simulation. Quite often, these may be modeled as probability distributions, either continuous or discrete.
2. Construct a simulation table. Each simulation table is different, for each is developed for the problem at hand. An example of a simulation table is shown in Table 2.1. In this example there are p inputs, x_{ij}, $j = 1, 2, \ldots, p$, and one response, y_i, for each of repetitions $i = 1, 2, \ldots, n$. Initialize the table by filling in the data for repetition 1.
3. For each repetition i, generate a value for each of the p inputs, and evaluate the function, calculating a value of the response y_i. The input values may be computed by sampling values from the distributions determined in step 1. A response typically depends on the inputs and one or more previous responses.

This chapter gives a number of simulation examples in queueing, inventory, and reliability. The two queueing examples provide a single-server and

two-server system, respectively. (Chapter 6 provides more insight into queueing models.) The first inventory example involves a problem that has a closed-form solution; thus the simulation solution can be compared to the mathematical solution. The second inventory example pertains to the classic order-level model.

Finally, there is an example that introduces the concept of random normal numbers and a model for the determinination of lead-time demand.

2.1 Simulation of Queueing Systems

A queueing system is described by its calling population, the nature of the arrivals, the service mechanism, the system capacity, and the queueing discipline. These attributes of a queueing system are described in detail in Chapter 6. A simple single-channel queueing system is portrayed in Figure 2.1.

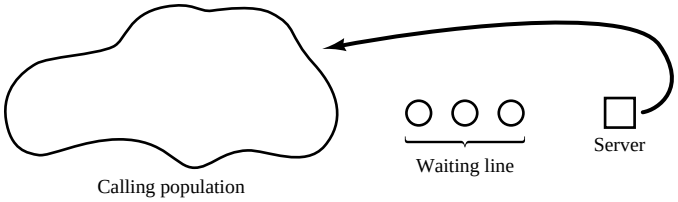


Figure 2.1 Queueing system.

In the single-channel queue, the calling population is infinite; that is, if a unit leaves the calling population and joins the waiting line or enters service, there is no change in the arrival rate of other units that may need service. Arrivals for service occur one at a time in a random fashion; once they join the waiting line, they are eventually served. In addition, service times are of some random length according to a probability distribution which does not change over time. The system capacity has no limit, meaning that any number of units can wait in line. Finally, units are served in the order of their arrival (often called FIFO: first in, first out) by a single server or channel.

Table 2.1 Simulation Table

	Inputs						*Response*
Repetitions	x_{i1}	x_{i2}	$\cdots$	x_{ij}	$\cdots$	x_{ip}	y_i
1							
2							
3							
.							
.							
.							
n							

Arrivals and services are defined by the distribution of the time between arrivals and the distribution of service times, respectively. For any simple single- or multi-channel queue, the overall effective arrival rate must be less than the total service rate, or the waiting line will grow without bound. When queues grow without bound, they are termed "explosive" or unstable. (In some reentrant queueing networks in which units return a number of times to the same server before finally exiting the system, the condition about arrival rate being less than service rate may not guarantee stability. See Harrison and Nguyen [1995] for more explanation. Interestingly, this type of instability was noticed first, not in theory, but in actual manufacturing in semiconductor plants.) More complex situations may occur—for example, arrival rates that are greater than service rates for short periods of time, or networks of queues with routing. However, this chapter sticks to the simplest, more basic queues.

Prior to introducing several simulations of queueing systems, it is necessary to understand the concepts of system state, events, and simulation clock. (These concepts are studied systematically in Chapter 3.) The *state* of the system is the number of units in the system and the status of the server, busy or idle. An *event* is a set of circumstances that cause an instantaneous change in the state of the system. In a single-channel queueing system there are only two possible events that can affect the state of the system. They are the entry of a unit into the system (the arrival event) or the completion of service on a unit (the departure event). The queueing system includes the server, the unit being serviced (if one is being serviced), and units in the queue (if any are waiting). The *simulation clock* is used to track simulated time.

If a unit has just completed service, the simulation proceeds in the manner shown in the flow diagram of Figure 2.2. Note that the server has only two possible states: it is either busy or idle.

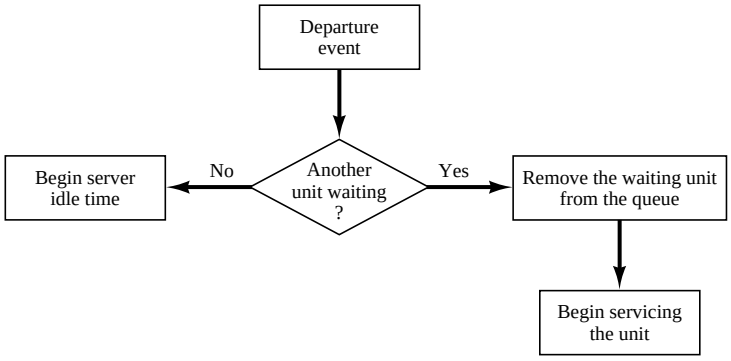


Figure 2.2 Service-just-completed flow diagram.

The arrival event occurs when a unit enters the system. The flow diagram for the arrival event is shown in Figure 2.3. The unit may find the server either idle or busy; therefore, either the unit begins service immediately, or it enters the queue for the server. The unit follows the course of action shown in Figure 2.4.

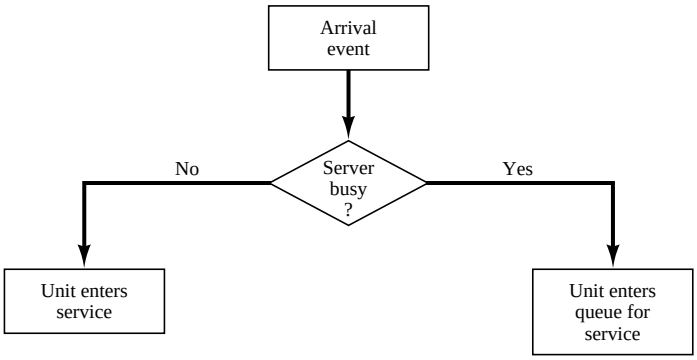


Figure 2.3 Unit-entering-system flow diagram.

If the server is busy, the unit enters the queue. If the server is idle and the queue is empty, the unit begins service. It is not possible for the server to be idle and the queue to be nonempty.

		Queue status	
		Not empty	Empty
Server status	Busy	Enter queue	Enter queue
	Idle	Impossible	Enter service

Figure 2.4 Potential unit actions upon arrival.

After the completion of a service the server may become idle or remain busy with the next unit. The relationship of these two outcomes to the status of the queue is shown in Figure 2.5. If the queue is not empty, another unit will enter the server and it will be busy. If the queue is empty, the server will be idle after a service is completed. These two possibilities are shown as the shaded portions of Figure 2.5. It is impossible for the server to become busy if the queue is empty when a service is completed. Similarly, it is impossible for the server to be idle after a service is completed when the queue is not empty.

		Queue status	
		Not empty	Empty
Server outcomes	Busy		Impossible
	Idle	Impossible	

Figure 2.5 Server outcomes after service completion.

Now, how can the events described above occur in simulated time? Simulations of queueing systems generally require the maintenance of an event list for determining what happens next. The event list tracks the future times

at which the different types of events occur. Simulations using event lists are described in Chapter 3. This chapter simplifies the simulation by tracking each unit explicitly. Simulation clock times for arrivals and departures are computed in a simulation table customized for each problem. In simulation, events usually occur at random times, the randomness imitating uncertainty in real life. For example, it is not known with certainty when the next customer will arrive at a grocery checkout counter, or how long the bank teller will take to complete a transaction. In these cases, a statistical model of the data is developed from either data collected and analyzed, or subjective estimates and assumptions.

The randomness needed to imitate real life is made possible through the use of "random numbers." Random numbers are distributed uniformly and independently on the interval (0, 1). Random digits are uniformly distributed on the set $\{0, 1, 2, \ldots, 9\}$. Random digits can be used to form random numbers by selecting the proper number of digits for each random number and placing a decimal point to the left of the value selected. The proper number of digits is dictated by the accuracy of the data being used for input purposes. If the input distribution has values with two decimal places, two digits are taken from a random-digits table (such as Table A.1) and the decimal point is placed to the left to form a random number.

Random numbers can also be generated in simulation packages and in spreadsheets such as Excel®. For example, Excel has a macro function called RAND() that returns a "random" number between 0 and 1. When numbers are generated using a procedure, they are often referred to as pseudo-random numbers. Since the method is known, it is always possible to know the sequence of numbers that will be generated prior to the simulation. The most commonly used methods for generating random numbers are discussed in Chapter 7.

In a single-channel queueing system interarrival times and service times are generated from the distributions of these random variables. The examples that follow show how such times are generated. For simplicity, assume that the times between arrivals were generated by rolling a die five times and recording the up face. Table 2.2 contains a set of five interarrival times generated in this manner. These five interarrival times are used to compute the arrival times of six customers at the queueing system.

Table 2.2 Interarrival and Clock Times

Customer	*Interarrival Time*	*Arrival Time on Clock*
1	–	0
2	2	2
3	4	6
4	1	7
5	2	9
6	6	15

Table 2.3 Service Times

Customer	*Service Time*
1	2
2	1
3	3
4	2
5	1
6	4

The first customer is assumed to arrive at clock time 0. This starts the clock in operation. The second customer arrives two time units later, at a clock time of 2. The third customer arrives four time units later, at a clock time of 6; and so on.

The second time of interest is the service time. Table 2.3 contains service times generated at random from a distribution of service times. The only possible service times are one, two, three, and four time units. Assuming that all four values are equally likely to occur, these values could have been generated by placing the numbers one through four on chips and drawing the chips from a hat with replacement, being sure to record the numbers selected. Now, the interarrival times and service times must be meshed to simulate the single-channel queueing system. As shown in Table 2.4, the first customer arrives at clock time 0 and immediately begins service, which requires two minutes. Service is completed at clock time 2. The second customer arrives at clock time 2 and is finished at clock time 3. Note that the fourth customer arrived at clock time 7, but service could not begin until clock time 9. This occurred because customer 3 did not finish service until clock time 9.

Table 2.4 was designed specifically for a single-channel queue which serves customers on a first-in, first-out (FIFO) basis. It keeps track of the clock time

Table 2.4 Simulation Table Emphasizing Clock Times

A	B	C	D	E
Customer Number	*Arrival Time (Clock)*	*Time Service Begins (Clock)*	*Service Time (Duration)*	*Time Service Ends (Clock)*
1	0	0	2	2
2	2	2	1	3
3	6	6	3	9
4	7	9	2	11
5	9	11	1	12
6	15	15	4	19

Table 2.5 Chronological Ordering of Events

Event Type	*Customer Number*	*Clock Time*
Arrival	1	0
Departure	1	2
Arrival	2	2
Departure	2	3
Arrival	3	6
Arrival	4	7
Departure	3	9
Arrival	5	9
Departure	4	11
Departure	5	12
Arrival	6	15
Departure	6	19

at which each event occurs. The second column of Table 2.4 records the clock time of each arrival event, while the last column records the clock time of each departure event. The occurrence of the two types of events in chronological order is shown in Table 2.5 and Figure 2.6.

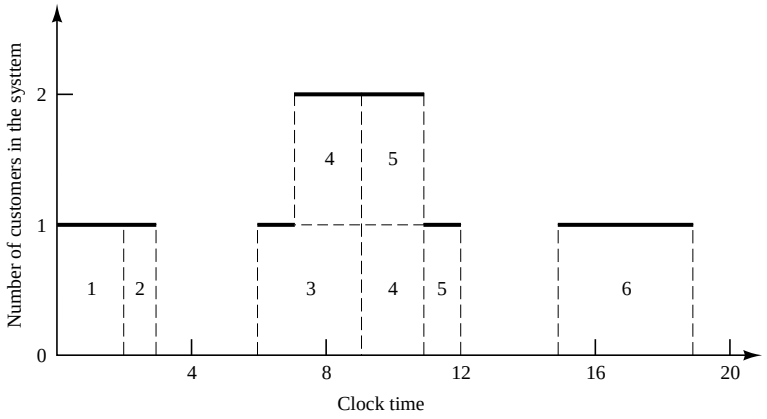


Figure 2.6 Number of customers in the system.

It should be noted that Table 2.5 is ordered by clock time, in which case the events may or may not be ordered by customer number. The chronological ordering of events is the basis of the approach to discrete-event simulation described in Chapter 3.

Figure 2.6 depicts the number of customers in the system at the various clock times. It is a visual image of the event listing of Table 2.5. Customer 1

Table 2.6 Distribution of Time Between Arrivals

Time between Arrivals (Minutes)	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
1	0.125	0.125	001–125
2	0.125	0.250	126–250
3	0.125	0.375	251–375
4	0.125	0.500	376–500
5	0.125	0.625	501–625
6	0.125	0.750	626–750
7	0.125	0.875	751–875
8	0.125	1.000	876–000

is in the system from clock time 0 to clock time 2. Customer 2 arrives at clock time 2 and departs at clock time 3. No customers are in the system from clock time 3 to clock time 6. During some time periods two customers are in the system, such as at clock time 8, when both customers 3 and 4 are in the system. Also, there are times when events occur simultaneously, such as at clock time 9, when customer 5 arrives and customer 3 departs.

Example 2.1 follows the logic described above while keeping track of a number of attributes of the system. Example 2.2 is concerned with a two-channel queueing system. The flow diagrams for a multichannel queueing system are slightly different from those for a single-channel system. The development and interpretation of these flow diagrams is left as an exercise for the reader.

EXAMPLE 2.1 Single-Channel Queue

A small grocery store has only one checkout counter. Customers arrive at this checkout counter at random from 1 to 8 minutes apart. Each possible value of interarrival time has the same probability of occurrence, as shown in Table 2.6. The service times vary from 1 to 6 minutes with the probabilities shown in Table 2.7. The problem is to analyze the system by simulating the arrival and service of 20 customers.

Table 2.7 Service-Time Distribution

Service Time (Minutes)	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
1	0.10	0.10	01–10
2	0.20	0.30	11–30
3	0.30	0.60	31–60
4	0.25	0.85	61–85
5	0.10	0.95	86–95
6	0.05	1.00	96–00

In actuality, 20 customers is too small a sample size to allow drawing any reliable conclusions. The accuracy of the results is enhanced by increasing the sample size, as discussed in Chapter 11. However, the purpose of the exercise is to demonstrate how simple simulations can be carried out in a table, either manually or with a spreadsheet, not to recommend changes in the grocery store. A second issue, discussed thoroughly in Chapter 11, is that of initial conditions. A simulation of a grocery store that starts with an empty system is not realistic unless the intention is to model the system from startup or to model until steady-state operation is reached. Here, to keep things simple, starting conditions and concerns are overlooked.

A set of uniformly distributed random numbers is needed to generate the arrivals at the checkout counter. Random numbers have the following properties:

1. The set of random numbers is uniformly distributed between 0 and 1.
2. Successive random numbers are independent.

With tabular simulations, random digits such as those found in Table A.1 in the Appendix can be converted to random numbers. If using a spreadsheet, most have a built-in random-number generator such as RAND() in Excel. The example in the text uses random digits from Table A.1; in some of the exercises the student is asked to use a spreadsheet.

Random digits are converted to random numbers by placing a decimal point appropriately. Since the probabilities in Table 2.6 are accurate to 3 significant digits, three-place random numbers will suffice. It is necessary to list only 19 random numbers to generate times between arrivals. Why only 19 numbers? The first arrival is assumed to occur at time 0, so only 19 more arrivals need to be generated to end up with 20 customers. Similarly, for Table 2.7, two-place random numbers will suffice.

The rightmost two columns of Tables 2.6 and 2.7 are used to generate random arrivals and random service times. The third column in each table contains the cumulative probability for the distribution. The rightmost column contains the random digit-assignment. In Table 2.6, the first random-digit assignment is 001–125. There are 1000 three-digit values possible (001 through 000). The probability of a time-between-arrivals of 1 minute is 0.125, and 125 of the 1000 random-digit values are assigned to such an occurrence. Times between arrivals for 19 customers are generated by listing 19 three-digit values from Table A.1 and comparing them to the random-digit assignment of Table 2.6.

For manual simulations, it is good practice to start at a random position in the random-digit table and proceed in a systematic direction, never re-using the same stream of digits in a given problem. If the same pattern is used repeatedly, bias could result, because the same event pattern would be generated. In Excel, each time the random function RAND() is evaluated, it returns a new random value.

The time-between-arrival determination is shown in Table 2.8. Note that the first random digits are 913. To obtain the corresponding time between

Table 2.8 Time-Between-Arrivals Determination

Customer	Random Digits	Time between Arrivals (Minutes)	Customer	Random Digits	Time between Arrivals (Minutes)
1	–	–	11	109	1
2	913	8	12	093	1
3	727	6	13	607	5
4	015	1	14	738	6
5	948	8	15	359	3
6	309	3	16	888	8
7	922	8	17	106	1
8	753	7	18	212	2
9	235	2	19	493	4
10	302	3	20	535	5

arrivals, enter the fourth column of Table 2.6 and read 8 minutes from the first column of the table. Alternatively, we see that 0.913 is between the cumulative probabilities 0.876 and 1.000, again resulting in 8 minutes as the generated time.

Service times for all 20 customers are shown in Table 2.9. These service times were generated based on the methodology described above, together with the aid of Table 2.7. The first customer's service time is 4 minutes because the random digits 84 fall in the bracket 61–85, or alternatively because the derived random number 0.84 falls between the cumulative probabilities 0.61 and 0.85.

Table 2.9 Service Times Generated

Customer	Random Digits	Service Time (Minutes)	Customer	Random Digits	Service Time (Minutes)
1	84	4	11	32	3
2	10	1	12	94	5
3	74	4	13	79	4
4	53	3	14	05	1
5	17	2	15	79	5
6	79	4	16	84	4
7	91	5	17	52	3
8	67	4	18	55	3
9	89	5	19	30	2
10	38	3	20	50	3

The essence of a manual simulation is the simulation table. These tables are designed for the problem at hand, with columns added to answer the questions posed. The simulation table for the single-channel queue, shown in Table 2.10, is an extension of the type of table already seen in Table 2.4. The first step is to initialize the table by filling in cells for the first customer. The first customer is assumed to arrive at time 0. Service begins immediately and finishes at time 4. The customer was in the system for 4 minutes. After the first customer, subsequent rows in the table are based on the random numbers for interarrival time and service time and the completion time of the previous customer. For example, the second customer arrives at time 8. Thus, the server (checkout person) was idle for 4 minutes. Skipping down to the fourth customer, it is seen that this customer arrived at time 15 but could not be served until time 18. This customer had to wait in the queue for 3 minutes. This process continues for all 20 customers. Extra columns have been added to collect statistical measures of performance such as each customer's time in the system and the server's idle time (if any) since the previous customer departed. In order to compute summary statistics, totals are formed as shown for service times, time customers spend in the system, idle time of the server, and time the customers wait in the queue.

In the exercises, the reader is asked to implement the simulation table for the single-channel queue, Table 2.10, in Excel or another spreadsheet. Here we give some hints when using Excel. The key column to compute is column E, the "Time Service Begins". (We leave for the reader the question of how to compute the random interarrival and service times, but suggest the RAND() random number generator or other built-in distribution in Excel.) First, the reader may fill in row 1 for the first customer manually. The values for the remaining customers must use macro formulas (which begin with an equals sign in Excel). Note that a customer begins service at the later of its own arrival time (column C) or the completion time (column G) of the previous customer. Therefore, for customer 10, service begins at E10 = MAX(C10, G9), where MAX() is the Excel macro function that returns the maximum value in a range or list of cells. This easily generalizes to other customers. (The statistical measures in columns H and I are easily computed by simple subtractions–also left for the reader.) A final hint on how to verify your spreadsheet model: instead of using a random function for arrivals and service times, type in the actual values given in Table 2.10 in columns B and D. If your formulas are correct, the spreadsheet should duplicate Table 2.10 exactly. After verification, replace the numbers by an appropriate random function. Then on each recalculation of the spreadsheet (function key F9 in Excel), it will generate new random numbers and you will get a new ''run" of the simulation.

Some of the findings from the simulation in Table 2.10 are as follows:

1. The average waiting time for a customer is 2.8 minutes. This is determined in the following manner:

$$\begin{aligned}\text{average waiting time (minutes)} &= \frac{\text{total time customers wait in queue (minutes)}}{\text{total numbers of customers}} \\ &= \frac{56}{20} = 2.8 \text{ minutes}\end{aligned}$$

2. The probability that a customer has to wait in the queue is 0.65. This is determined in the following manner:

$$\begin{aligned}\text{probability (wait)} &= \frac{\text{number of customers who wait}}{\text{total number of customers}} \\ &= \frac{13}{20} = 0.65\end{aligned}$$

3. The fraction of idle time of the server is 0.21. This is determined in the following manner:

$$\begin{aligned}\text{probability of idle server} &= \frac{\text{total idle time of server (minutes)}}{\text{total run time of simulation (minutes)}} \\ &= \frac{18}{86} = 0.21\end{aligned}$$

The probability of the server being busy is the complement of 0.21, or 0.79.

4. The average service time is 3.4 minutes, determined as follows:

$$\begin{aligned}\text{average service time (minutes)} &= \frac{\text{total service time (minutes)}}{\text{total number of customers}} \\ &= \frac{68}{20} = 3.4 \text{ minutes}\end{aligned}$$

This result can be compared with the expected service time by finding the mean of the service-time distribution using the equation

$$E(S) = \sum_{s=0}^{\infty} sp(s)$$

Applying the expected-value equation to the distribution in Table 2.7 gives an expected service time of:

$$= 1(0.10) + 2(0.20) + 3(0.30) + 4(0.25) + 5(0.10) + 6(0.05)$$

$$= 3.2 \text{ minutes}$$

The expected service time is slightly lower than the average service time in the simulation. The longer the simulation, the closer the average will be to $E(S)$.

Table 2.10 Simulation Table for Queueing Problem

A *Customer*	B *Time Since Last Arrival (Minutes)*	C *Arrival Time*	D *Service Time (Minutes)*	E *Time Service Begins*	F *Time Customer Waits in Queue (Minutes)*	G *Time Service Ends*	H *Time Customer Spends in System (Minutes)*	I *Idle Time of Server (Minutes)*
1	–	0	4	0	0	4	4	0
2	8	8	1	8	0	9	1	4
3	6	14	4	14	0	18	4	5
4	1	15	3	18	3	21	6	0
5	8	23	2	23	0	25	2	2
6	3	26	4	26	0	30	4	1
7	8	34	5	34	0	39	5	4
8	7	41	4	41	0	45	4	2
9	2	43	5	45	2	50	7	0
10	3	46	3	50	4	53	7	0
11	1	47	3	53	6	56	9	0
12	1	48	5	56	8	61	13	0
13	5	53	4	61	8	65	12	0
14	6	59	1	65	6	66	7	0
15	3	62	5	66	4	71	9	0
16	8	70	4	71	1	75	5	0
17	1	71	3	75	4	78	7	0
18	2	73	3	78	5	81	8	0
19	4	77	2	81	4	83	6	0
20	5	82	3	83	1	86	4	0
			68		56		124	18

5. The average time between arrivals is 4.3 minutes. This is determined in the following manner:

$$\begin{aligned}\text{average time between arrivals (minutes)} &= \frac{\text{sum of all times between arrivals (minutes)}}{\text{number of arrivals} - 1} \\ &= \frac{82}{19} = 4.3 \text{ minutes}\end{aligned}$$

One is subtracted from the denominator because the first arrival is assumed to occur at time 0. This result can be compared to the expected time between arrivals by finding the mean of the discrete uniform distribution whose endpoints are $a = 1$ and $b = 8$. The mean is given by

$$E(A) = \frac{a + b}{2} = \frac{1 + 8}{2} = 4.5 \text{ minutes}$$

The expected time between arrivals is slightly higher than the average. However, as the simulation becomes longer, the average value of the time between arrivals will approach the theoretical mean, $E(A)$.

6. The average waiting time of those who wait is 4.3 minutes. This is determined in the following manner:

$$\begin{aligned}\text{Average waiting time of those who wait (minutes)} &= \frac{\text{total time customers wait in queue (minutes)}}{\text{total number of customers who wait}} \\ &= \frac{56}{13} = 4.3 \text{ minutes}\end{aligned}$$

7. The average time a customer spends in the system is 6.2 minutes. This can be determined in two ways. First, the computation can be achieved by the following relationship:

$$\begin{aligned}\text{average time customer spends in the system (minutes)} &= \frac{\text{total time customers spend in the system (minutes)}}{\text{total number of customers}} \\ &= \frac{124}{20} = 6.2 \text{ minutes}\end{aligned}$$

The second way of computing this same result is to realize that the following relationship must hold:

$$\begin{aligned}&\text{average time customer spends in the system (minutes)} \\ &= \text{average time customer spends waiting in the queue (minutes)} \\ &+ \text{average time customer spends in service (minutes)}\end{aligned}$$

From findings 1 and 4 this results in:

Average time customer spends in the system (minutes)

$$= 2.8 + 3.4 = 6.2 \text{ minutes}$$

A decision maker would be interested in results of this type, but a longer simulation would increase the accuracy of the findings. However, some subjective inferences can be drawn at this point. Most customers have to wait; however, the average waiting time is not excessive. The server does not have an undue amount of idle time. Objective statements about the results would depend on balancing the cost of waiting with the cost of additional servers. (Simulations requiring variations of the arrival and service distributions, as well as implementation in a spreadsheet, are presented as exercises for the reader.) ▲

EXAMPLE 2.2 The Able Baker Carhop Problem

This example illustrates the simulation procedure when there is more than one service channel. Consider a drive-in restaurant where carhops take orders and bring food to the car. Cars arrive in the manner shown in Table 2.11. There are two carhops—Able and Baker. Able is better able to do the job and works a bit faster than Baker. The distribution of their service times is shown in Tables 2.12 and 2.13.

Table 2.11 Interarrival Distribution of Cars

Time between Arrivals (Minutes)	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
1	0.25	0.25	01–25
2	0.40	0.65	26–65
3	0.20	0.85	66–85
4	0.15	1.00	86–00

The simulation proceeds in a manner similar to Example 2.1, except that it is more complex because of the two servers. A simplifying rule is that Able gets the customer if both carhops are idle. Perhaps, Able has seniority. (The

Table 2.12 Service Distribution of Able

Service Time (Minutes)	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
2	0.30	0.30	01–30
3	0.28	0.58	31–58
4	0.25	0.83	59–83
5	0.17	1.00	84–00

Table 2.13 Service Distribution of Baker

Service Time (Minutes)	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
3	0.35	0.35	01–35
4	0.25	0.60	36–60
5	0.20	0.80	61–80
6	0.20	1.00	81–00

solution would be different if the decision were made at random or by any other rule.)

The problem is to find how well the current arrangement is working. To estimate the system measures of performance, a simulation of 1 hour of operation is made. A longer simulation would yield more reliable results, but for purposes of illustration a l-hour period has been selected.

The simulation proceeds in a manner similar to Example 2.1. Here there are more events: a customer arrives, a customer begins service from Able, a customer completes service from Able, a customer begins service from Baker, and a customer completes service from Baker. The simulation table is shown in Table 2.14.

In later exercises, the reader is asked to implement the simulation table, Table 2.14, in a spreadsheet such as Excel. Here we provide a few hints (and rules!). The row for the first customer is filled in manually, with the random-number function RAND() or another random function replacing the random digits. After the first customer, the cells for the other customers must be based on logic and formulas. For example, the "Clock Time of Arrival" (column D) in the row for the second customer is computed as follows:

```
D2 = D1 + C2
```

using notation similar to that used by most spreadsheets. (C2 is the time between arrivals 1 and 2.) This formula is easily generalized for any customer.

The logic to compute who gets a given customer, and when that service begins, is more complex. Here we give a hint using the Excel macro function IF(), which returns one of two values depending on whether a condition is true or false. [The syntax is IF(condition, value_if_true, value_if_false).] The logic goes as follows when a customer arrives: If the customer finds Able idle, the customer begins service immediately with Able. If Able is not idle but Baker is, then the customer begins service immediately with Baker. If both are busy, the

customer begins service with the first server to become free. The logic requires that we compute when Able and Baker will become free, for which we use the built-in Excel function for maximum over a range, MAX(). For example, for customer 10, Able will become free at MAX(H$1:H9), since service completion time is in column H and we need to look at customers 1–9. (Using H$1 instead of H1 works better with Excel when formulas are copied. The dollar sign indicates an absolute reference versus a relative reference to a cell.) The resulting formula to compute whether and when Able serves customer 10 is as follows:

```
F10 = IF(D10>MAX(H$1:H9),D10, IF(D10>MAX(K$1:K9),'''',
   MIN(MAX(H$1:H9),MAX(K$1:K9))))
```

In this formula, note that if the first condition (Able idle when customer 10 arrives) is true, then the customer begins immediately at the arrival time in D10. Otherwise, a second IF() function is evaluated, which says if Baker is idle, put nothing ("") in the cell. Otherwise, the function returns the time that Able or Baker becomes idle, whichever is first [the minimum or MIN() of their respective completion times]. A similar formula applies to cell I10 for "Time Service Begins" for Baker. For service times for Able, you could use another IF() function to make the cell blank or have a value:

```
G10 = IF(F10 > 0,new_service_time, "")
H10 = IF(F10 > 0, F10+G10, "")
```

and similarly for Baker. With these hints, we leave the formula for `new_service_time` as well as the remainder of the solution to the reader.

The analysis of Table 2.14 results in the following:

1. Over the 62-minute period Able was busy 90% of the time.
2. Baker was busy only 69% of the time. The seniority rule keeps Baker less busy (and gives Able more tips).
3. Nine of the 26 arrivals (about 35%) had to wait. The average waiting time for all customers was only about 0.42 minute (25 seconds), which is very small.
4. Those nine who did have to wait only waited an average of 1.22 minutes, which is quite low.
5. In summary, this system seems well balanced. One server cannot handle all the diners, and three servers would probably be too many. Adding an additional server would surely reduce the waiting time to nearly zero. However, the cost of waiting would have to be quite high to justify an additional server. ▲

Table 2.14 Simulation Table for Carhop Example

A	B	C	D	E	F	G	H	I	J	K	L
					Able			*Baker*			
Customer No.	*Random Digits for Arrival*	*Time between Arrivals*	*Clock Time of Arrival*	*Random Digits for Service*	*Time Service Begins*	*Service Time*	*Time Service Ends*	*Time Service Begins*	*Service Time*	*Time Service Ends*	*Time in Queue*
1	–	–	0	95	0	5	5				0
2	26	2	2	21				2	3	5	0
3	98	4	6	51	6	3	9				0
4	90	4	10	92	10	5	15				0
5	26	2	12	89				12	6	18	0
6	42	2	14	38	15	3	18				1
7	74	3	17	13	18	2	20				1
8	80	3	20	61	20	4	24				0
9	68	3	23	50				23	4	27	0
10	22	1	24	49	24	3	27				0
11	48	2	26	39	27	3	30				1
12	34	2	28	53				28	4	32	0
13	45	2	30	88	30	5	35				0
14	24	1	31	01				32	3	35	1
15	34	2	33	81	35	4	39				2
16	63	2	35	53				35	4	39	0
17	38	2	37	81	39	4	43				2
18	80	3	40	64				40	5	45	0
19	42	2	42	01	43	2	45				1
20	56	2	44	67	45	4	49				1
21	89	4	48	01				48	3	51	0
22	18	1	49	47	49	3	52				0
23	51	2	51	75				51	5	56	0
24	71	3	54	57	54	3	57				0
25	16	1	55	87				56	6	62	1
26	92	4	59	47	59	3	62				0
						56			43		11

2.2 Simulation of Inventory Systems

An important class of simulation problems involves inventory systems. A simple inventory system is shown in Figure 2.7. This inventory system has a periodic review of length N, at which time the inventory level is checked. An order is made to bring the inventory up to the level M. At the end of the first review period, an order quantity, Q_1, is placed. In this inventory system the lead time (i.e., the length of time between the placement and receipt of an order) is zero. Since demands are not usually known with certainty, the order quantities are probabilistic. Demand is shown as being uniform over the time period in Figure 2.7. In actuality, demands are not usually uniform and do fluctuate over time. One possibility is that demands all occur at the beginning of the cycle. Another is that the lead time is random of some positive length.

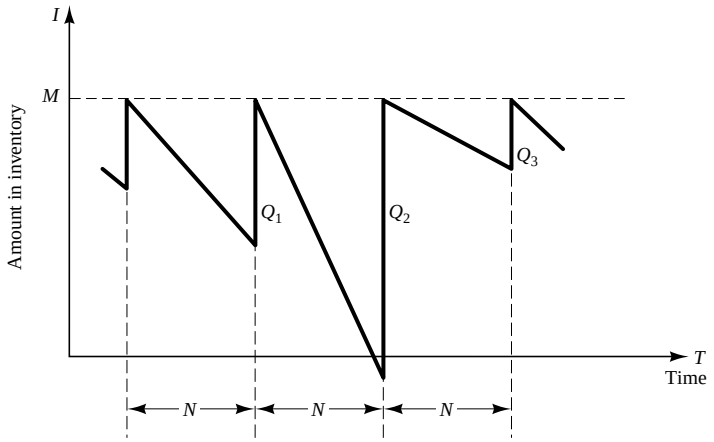


Figure 2.7 Probabilistic order-level inventory system.

Notice that in the second cycle, the amount in inventory drops below zero, indicating a shortage. In Figure 2.7, these units are backordered; when the order arrives, the demand for the backordered items is satisfied first. To avoid shortages, a buffer, or safety, stock would need to be carried.

Carrying stock in inventory has an associated cost attributed to the interest paid on the funds borrowed to buy the items (this also could be considered as the loss from not having the funds available for other investment purposes). Other costs can be placed in the carrying or holding cost column: renting of storage space, hiring guards, and so on. An alternative to carrying high inventory is to make more frequent reviews, and consequently, more frequent purchases or replenishments. This has an associated cost: the ordering cost. Also, there is a cost in being short. Customers may get angry, with a subsequent loss of good will. Larger inventories decrease the possibilities of shortages. These costs must be traded off in order to minimize the total cost of an inventory system.

The total cost (or total profit) of an inventory system is the measure of performance. This can be affected by the policy alternatives. For example,

in Figure 2.7, the decision maker can control the maximum inventory level, M, and the length of the cycle, N. What effect does changing N have on the various costs?

In an (M, N) inventory system, the events that may occur are: the demand for items in the inventory, the review of the inventory position, and the receipt of an order at the end of each review period. When the lead time is zero, as in Figure 2.7, the last two events occur simultaneously.

In the following example for deciding how many newspapers to buy, only a single time period of specified length is relevant and only a single procurement is made. Inventory remaining at the end of the single time period is sold for scrap or discarded. A wide variety of real-world problems are of this form, including the stocking of spare parts, perishable items, style goods, and special seasonal items [Hadley and Whitin, 1963].

EXAMPLE 2.3 The Newspaper Seller's Problem

A classical inventory problem concerns the purchase and sale of newspapers. The paper seller buys the papers for 33 cents each and sells them for 50 cents each. Newspapers not sold at the end of the day are sold as scrap for 5 cents each. Newspapers can be purchased in bundles of 10. Thus, the paper seller can buy 50, 60, and so on. There are three types of newsdays, "good," "fair," and "poor," with probabilities of 0.35, 0.45, and 0.20, respectively. The distribution of papers demanded on each of these days is given in Table 2.15. The problem is to determine the optimal number of papers the newspaper seller should purchase. This will be accomplished by simulating demands for 20 days and recording profits from sales each day.

The profits are given by the following relationship:

$$\text{Profit} = \begin{pmatrix}\text{revenue}\\ \text{from sales}\end{pmatrix} - \begin{pmatrix}\text{cost of}\\ \text{newspapers}\end{pmatrix} - \begin{pmatrix}\text{lost profit from}\\ \text{excess demand}\end{pmatrix} + \begin{pmatrix}\text{salvage from sale}\\ \text{of scrap papers}\end{pmatrix}$$

Table 2.15 Distribution of Newspapers Demanded

	Demand Probability Distribution		
Demand	*Good*	*Fair*	*Poor*
40	0.03	0.10	0.44
50	0.05	0.18	0.22
60	0.15	0.40	0.16
70	0.20	0.20	0.12
80	0.35	0.08	0.06
90	0.15	0.04	0.00
100	0.07	0.00	0.00

Table 2.16 Random-Digit Assignment for Type of Newsday

Type of Newsday	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
Good	0.35	0.35	01–35
Fair	0.45	0.80	36–80
Poor	0.20	1.00	81–00

From the problem statement, the revenue from sales is 50 cents for each paper sold. The cost of newspapers is 33 cents for each paper purchased. The lost profit from excess demand is 17 cents for each paper demanded that could not be provided. Such a shortage cost is somewhat controversial but makes the problem much more interesting. The salvage value of scrap papers is 5 cents each.

Tables 2.16 and 2.17 provide the random-digit assignments for the types of newsdays and the demands for those newsdays. To solve this problem by simulation requires setting a policy of buying a certain number of papers each day, then simulating the demands for papers over the 20-day time period to determine the total profit. The policy (number of newspapers purchased) is changed to other values and the simulation repeated until the best value is found.

Table 2.17 Random-Digit Assignments for Newspapers Demanded

	Cumulative Distribution			*Random-Digit Assignment*		
Demand	*Good*	*Fair*	*Poor*	*Good*	*Fair*	*Poor*
40	0.03	0.10	0.44	01–03	01–10	01–44
50	0.08	0.28	0.66	04–08	11–28	45–66
60	0.23	0.68	0.82	09–23	29–68	67–82
70	0.43	0.88	0.94	24–43	69–88	83–94
80	0.78	0.96	1.00	44–78	89–96	95–00
90	0.93	1.00	1.00	79–93	97–00	
100	1.00	1.00	1.00	94–00		

The simulation table for the decision to purchase 70 newspapers is shown in Table 2.18.

On day 1 the demand is for 60 newspapers. The revenue from the sale of 60 newspapers is \$30.00. Ten newspapers are left over at the end of the day. The salvage value at 5 cents each is 50 cents. The profit for the first day is determined as follows:

$$\text{Profit} = \$30.00 - \$23.10 - 0 + \$.50 = \$7.40$$

Table 2.18 Simulation Table for Purchase of 70 Newspapers

Day	*Random Digits for Type of Newsday*	*Type of Newsday*	*Random Digits for Demand*	*Demand*	*Revenue from Sales*	*Lost Profit from Excess Demand*	*Salvage from Sale of Scrap*	*Daily Profit*
1	94	Poor	80	60	$30.00	–	$0.50	$7.40
2	77	Fair	20	50	25.00	–	1.00	2.90
3	49	Fair	15	50	25.00	–	1.00	2.90
4	45	Fair	88	70	35.00	–	–	11.90
5	43	Fair	98	90	35.00	$3.40	–	8.50
6	32	Good	65	80	35.00	1.70	–	10.20
7	49	Fair	86	70	35.00	–	–	11.90
8	00	Poor	73	60	30.00	–	0.50	7.40
9	16	Good	24	70	35.00	–	–	11.90
10	24	Good	60	80	35.00	1.70	–	10.20
11	31	Good	60	80	35.00	1.70	–	10.20
12	14	Good	29	70	35.00	–	–	11.90
13	41	Fair	18	50	25.00	–	1.00	2.90
14	61	Fair	90	80	35.00	1.70	–	10.20
15	85	Poor	93	70	35.00	–	–	11.90
16	08	Good	73	80	35.00	1.70	–	10.20
17	15	Good	21	60	30.00	–	0.50	7.40
18	97	Poor	45	50	25.00	–	1.00	2.90
19	52	Fair	76	70	35.00	–	–	11.90
20	78	Fair	96	80	35.00	1.70	–	10.20
					$645.00	$13.60	$5.50	$174.90

On the fifth day the demand is greater than the supply. The revenue from sales is $35.00, since only 70 papers are available under this policy. An additional 20 papers could have been sold. Thus, a lost profit of $3.40 (20 × 17 cents) is assessed. The daily profit is determined as follows:

$$\text{Profit} = \$35.00 - \$23.10 - \$3.40 + 0 = \$8.50$$

The profit for the 20-day period is the sum of the daily profits, $174.90. It can also be computed from the totals for the 20 days of the simulation as follows:

$$\text{Total profit} = \$645.00 - \$462.00 - \$13.60 + \$5.50 = \$174.90$$

In general, since the results of one day are independent of those of previous days, inventory problems of this type are easier than queueing problems when solved in a spreadsheet such as Excel. The determination of the optimal number of newspapers to purchase is left as an exercise for the reader. ▲

EXAMPLE 2.4 Simulation of an (M, N) Inventory System

This example follows the pattern of the probabilistic order-level inventory system shown in Figure 2.7. Suppose that the maximum inventory level, M, is 11 units and the review period, N, is 5 days. The problem is to estimate, by simulation, the average ending units in inventory and the number of days when a shortage condition occurs. The distribution of the number of units demanded per day is shown in Table 2.19. In this example, lead time is a random variable, as shown in Table 2.20. Assume that orders are placed at the close of business and are received for inventory at the beginning of business as determined by the lead time.

Table 2.19 Random-Digit Assignments for Daily Demand

Demand	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
0	0.10	0.10	01–10
1	0.25	0.35	11–35
2	0.35	0.70	36–70
3	0.21	0.91	71–91
4	0.09	1.00	92–00

Table 2.20 Random-Digit Assignments for Lead Time

Lead Time (Days)	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
1	0.6	0.6	1–6
2	0.3	0.9	7–9
3	0.1	1.0	0

Table 2.21 Simulation Tables for (M, N) Inventory System

Cycle	*Day*	*Beginning Inventory*	*Random Digits for Demand*	*Demand*	*Ending Inventory*	*Shortage Quantity*	*Order Quantity*	*Random Digits for Lead Time*	*Days until Order Arrives*
1	1	3	24	1	2	0	–	–	1
	2	2	35	1	1	0	–	–	0
	3	9	65	2	7	0	–	–	–
	4	7	81	3	4	0	–	–	–
	5	4	54	2	2	0	9	5	1
2	1	2	03	0	2	0	–	–	0
	2	11	87	3	8	0	–	–	–
	3	8	27	1	7	0	–	–	–
	4	7	73	3	4	0	–	–	–
	5	4	70	2	2	0	9	0	3
3	1	2	47	2	0	0	–	–	2
	2	0	45	2	0	2	–	–	1
	3	0	48	2	0	4	–	–	0
	4	9	17	1	4	0	–	–	–
	5	4	09	0	4	0	7	3	1
4	1	4	42	2	2	0	–	–	0
	2	9	87	3	6	0	–	–	–
	3	6	26	1	5	0	–	–	–
	4	5	36	2	3	0	–	–	–
	5	3	40	2	1	0	10	4	1
5	1	1	07	0	1	0	–	–	0
	2	11	63	2	9	0	–	–	–
	3	9	19	1	8	0	–	–	–
	4	8	88	3	5	0	–	–	–
	5	5	94	4	1	0	10	8	2
					88				

To make an estimate of the mean units in ending inventory, many cycles would have to be simulated. For purposes of this example, only five cycles will be shown. The reader is asked to continue the example as an exercise at the end of the chapter.

The random-digit assignments for daily demand and lead time are shown in the rightmost columns of Tables 2.19 and 2.20. The resulting simulation table is shown in Table 2.21. The simulation has been started with the inventory level at 3 units and an order of 8 units scheduled to arrive in 2 days' time.

Following the simulation table for several selected days indicates how the process operates. The order for 8 units is available on the morning of the third day of the first cycle, raising the inventory level from 1 unit to 9 units. Demands during the remainder of the first cycle reduced the ending inventory level to 2 units on the fifth day. Thus, an order for 9 units was placed. The lead time for this order was 1 day. The order of 9 units was added to inventory on the morning of day 2 of cycle 2.

Notice that the beginning inventory on the second day of the third cycle was zero. An order for 2 units on that day led to a shortage condition. The units were backordered on that day and the next day also. On the morning of day 4 of cycle 3 there was a beginning inventory of 9 units. The 4 units that were backordered and the 1 unit demanded that day reduced the ending inventory to 4 units.

Based on five cycles of simulation, the average ending inventory is approximately 3.5 ($88 \div 25$) units. On 2 of 25 days a shortage condition existed. ▲

2.3 Other Examples of Simulation

This section includes examples of the simulation of a reliability problem, a bombing mission, and the generation of the lead-time demand distribution given the distributions of demand and lead time.

EXAMPLE 2.5 A Reliability Problem

A large milling machine has three different bearings that fail in service. The cumulative distribution function of the life of each bearing is identical, as shown in Table 2.22. When a bearing fails, the mill stops, a repairperson is called, and a new bearing is installed. The delay time of the repairperson's arriving at the milling machine is also a random variable, with the distribution given in Table 2.23. Downtime for the mill is estimated at $5 per minute. The direct on-site cost of the repairperson is $15 per hour. It takes 20 minutes to change one bearing, 30 minutes to change two bearings, and 40 minutes to change three bearings. The bearings cost $16 each. A proposal has been made to replace all three bearings whenever a bearing fails. Management needs an evaluation of this proposal.

Table 2.24 represents a simulation of 20,000 hours of operation under the current method of operation. Note that there are instances where more than one bearing fails at the same time. This is unlikely to occur in practice and is

Table 2.22 Bearing-Life Distribution

Bearing Life (Hours)	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
1000	0.10	0.10	01–10
1100	0.13	0.23	11–23
1200	0.25	0.48	24–48
1300	0.13	0.61	49–61
1400	0.09	0.70	62–70
1500	0.12	0.82	71–82
1600	0.02	0.84	83–84
1700	0.06	0.90	85–90
1800	0.05	0.95	91–95
1900	0.05	1.00	96–00

due to using a rather coarse grid of 100 hours. It will be assumed in this example that the times are never exactly the same, and thus no more than one bearing is changed at any breakdown. Sixteen bearing changes were made for bearings 1 and 2, but only 14 bearing changes were required for bearing 3. The cost of the current system is estimated as follows:

Cost of bearings = 46 bearings × \$16/bearing = \$736

Cost of delay time = (110 + 125 + 95) minutes × \$5/minute = \$1650

Cost of downtime during repair =

46 bearings × 20 minutes/bearing × \$5/minute = \$4600

Cost of repairpersons =

46 bearings × 20 minutes/bearing × \$15/60 minutes= \$230

Total cost = \$736 + \$1650 + \$4600 + \$230 = \$7216

Table 2.23 Delay-Time Distribution

Delay Time (Minutes)	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
5	0.6	0.6	1–6
10	0.3	0.9	7–9
15	0.1	1.0	0

Table 2.25 is a simulation using the proposed method. Notice that bearing life is taken from Table 2.24, so that for as many bearings as were used in the current method, the bearing life is identical for both methods.

Table 2.24 Bearing Replacement Using Current Method

	Bearing 1					*Bearing 2*					*Bearing 3*				
	RD[a]	*Life (Hours)*	*Accumulated Life (Hours)*	*RD*	*Delay (Minutes)*	*RD*	*Life (Hours)*	*Accumulated Life (Hours)*	*RD*	*Delay (Minutes)*	*RD*	*Life (Hours)*	*Accumulated Life (Hours)*	*RD*	*Delay (Minutes)*
1	67	1,400	1,400	2	5	70	1,500	1,500	0	15	76	1,500	1,500	0	15
2	08	1,000	2,400	3	5	43	1,200	2,700	7	10	65	1,400	2,900	2	5
3	49	1,300	3,700	1	5	86	1,700	4,400	3	5	61	1,400	4,300	7	10
4	84	1,600	5,300	7	10	93	1,800	6,200	1	5	96	1,900	6,200	1	5
5	44	1,200	6,500	8	10	81	1,600	7,800	2	5	65	1,400	7,600	3	5
6	30	1,200	7,700	1	5	44	1,200	9,000	8	10	56	1,300	8,900	3	5
7	10	1,000	8,700	2	5	19	1,100	10,100	1	5	11	1,100	10,000	6	5
8	63	1,400	10,100	8	10	51	1,300	11,400	1	5	86	1,700	11,700	3	5
9	02	1,000	11,100	3	5	45	1,300	12,700	7	10	57	1,300	13,000	1	5
10	02	1,000	12,100	8	10	12	1,100	13,800	8	5	49	1,300	14,300	4	5
11	77	1,500	13,600	7	10	48	1,300	15,100	0	15	36	1,200	15,500	8	10
12	59	1,300	14,900	5	5	09	1,000	16,100	8	10	44	1,200	16,700	2	5
13	23	1,100	16,000	5	5	44	1,200	17,300	1	5	94	1,800	18,500	1	5
14	53	1,300	17,300	9	10	46	1,200	18,500	2	5	78	1,500	20,000	7	10
15	85	1,700	19,000	6	5	40	1,200	19,700	8	10					
16	75	1,500	20,500	4	5	52	1,300	21,000	5	5					
					110					125					95

[a] RD, random digits.

Table 2.25 Bearing Replacement using Proposed Method

	Bearing 1 Life (Hours)	*Bearing 2 Life (Hours)*	*Bearing 3 Life (Hours)*	*First Failure (Hours)*	*Accumulated Life (Hours)*	*RD*	*Delay (Minutes)*
1	1,400	1,500	1,500	1,400	1,400	3	5
2	1,000	1,200	1,400	1,000	2,400	7	10
3	1,300	1,700	1,400	1,300	3,700	5	5
4	1,600	1,800	1,900	1,600	5,300	1	5
5	1,200	1,600	1,400	1,200	6,500	4	5
6	1,200	1,200	1,300	1,200	7,700	3	5
7	1,000	1,100	1,100	1,000	8,700	7	10
8	1,400	1,300	1,700	1,300	10,000	8	10
9	1,000	1,300	1,300	1,000	11,000	8	10
10	1,000	1,100	1,300	1,000	12,000	3	5
11	1,500	1,300	1,200	1,200	13,200	2	5
12	1,300	1,000	1,200	1,000	14,200	4	5
13	1,100	1,200	1,800	1,100	15,300	1	5
14	1,300	1,200	1,500	1,200	16,500	6	5
15	1,700	1,200	63/1,400	1,200	17,700	2	5
16	1,500	1,300	21/1,100	1,100	18,800	7	10
17	85/1,700	53/1,300	23/1,100	1,100	19,900	0	15
18	05/1,000	29/1,200	51/1,300	1,000	20,900	5	5
							125

It is assumed that the bearings are in order on a shelf and they are taken sequentially and placed on the mill. Since the proposed method uses more bearings than the current method, the second simulation uses new random digits for generating the additional lifetimes. (When comparing two scenarios, the effect of using different random numbers versus common random numbers is discussed in Chapter 12.) The random digits that lead to the lives of the additional bearings are shown above the slashed line beginning with the 15th replacement of bearing 3. When the new policy is used, some 18 sets of bearings were required. In the two simulations, repairperson delays were not duplicated but were generated independently using different random digits. The total cost of the new policy is computed as follows:

$$\text{Cost of bearings} = 54 \text{ bearings} \times \$16/\text{bearing} = \$864$$

$$\text{Cost of delay time} = 125 \text{ minutes} \times \$5/\text{minute} = \$625$$

$$\text{Cost of downtime during repairs} =$$

$$18 \text{ sets} \times 40 \text{ minutes/set} \times \$5/\text{minute} = \$3600$$

$$\text{Cost of repairpersons} =$$

$$18 \text{ sets} \times 40 \text{ minutes/set} \times \$15/60 \text{ minutes} = \$180$$

$$\text{Total cost} = \$864 + \$625 + \$3600 + \$180 = \$5269$$

The new policy generates a savings of $1947 over a 20,000-hour simulation. If the machine runs continuously, the simulated time is about $2\frac{1}{4}$ years. Thus, the savings are about $865 per year. ▲

EXAMPLE 2.6 Random Normal Numbers

A classic simulation problem is that of a squadron of bombers attempting to destroy an ammunition depot shaped as shown in Figure 2.8. If a bomb lands anywhere on the depot, a hit is scored. Otherwise, the bomb is a miss. The aircraft fly in the horizontal direction. Ten bombers are in each squadron. The aiming point is the dot located in the heart of the ammunition dump. The point of impact is assumed to be normally distributed around the aiming point with a standard deviation of 600 meters in the horizontal direction and 300 meters in the vertical direction. The problem is to simulate the operation and make statements about the number of bombs on target.

Recall that the standardized normal variate, Z, with mean 0 and standard deviation 1, is distributed as

$$Z = \frac{X - \mu}{\sigma}$$

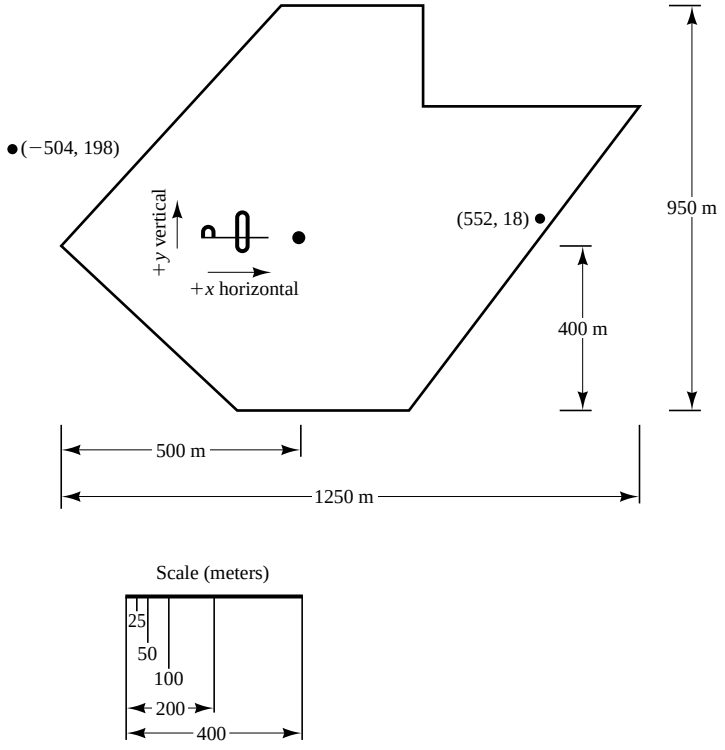


Figure 2.8 Ammunition depot.

where X is a normal random variable, μ is the true mean of the distribution of X, and σ is the standard deviation of X. Then,

$$X = Z\sigma + \mu.$$

In this example the aiming point can be considered as $(0, 0)$; that is, the μ value in the horizontal direction is 0, and similarly for the μ value in the vertical direction. Then,

$$X = Z\sigma_X$$
$$Y = Z\sigma_Y$$

where (X, Y) are the simulated coordinates of the bomb after it has fallen. Now, $\sigma_X = 600$ and $\sigma_Y = 300$. Therefore,

$$X = 600Z_i$$
$$Y = 300Z_j$$

The i and j subscripts have been added to indicate that the values of Z should be different. What are these Z values and where can they be found? The values of Z are random normal numbers. These can be generated from uniformly distributed random numbers, as discussed in Chapter 7. Alternatively, tables of random normal numbers have been generated. A small sample of random normal numbers is given in Table A.2. If using a spreadsheet, there is a built-in function that generates normal random numbers. (For Excel, use the Random Number Generation tool in the Analysis TookPak Add-In to generate any number of normal random values in a range of cells.)

To understand what happens in these bombing missions, a simulation of perhaps 10 or 20 runs might be conducted. However, space limitations do not permit such an extensive simulation. An example of one run will indicate how the simulation is performed. The table of random normal numbers is used in the same way as the table of random numbers: that is, start at a random place in the table and proceed in a systematic direction, avoiding overlap. Table 2.26 shows the results of a simulated run.

The mnemonic RNN_x stands for "random normal number to compute the x coordinate" and corresponds to Z_i above. The first random normal number used was -0.84, generating an x coordinate $600(-0.84) = -504$. The random normal number to generate the y coordinate was 0.66, resulting in a y coordinate of 198. Taken together, $(-504, 198)$ is a miss, for it is off the target. The resulting point and that of the third bomber are plotted on Figure 2.8. The 10 bombers had 3 hits and 7 misses. Many more runs are needed to assess the potential for destroying the dump. Additional runs appear as an exercise for the reader. This is an example of a Monte Carlo, or static, simulation, since time is not an element of the solution. ▲

Table 2.26 Simulated Bombing Run

		x Coordinate		*y Coordinate*	
Bomber	RNN_x	(600 RNN_x)	RNN_y	(300 RNN_y)	*Result*[a]
1	−0.84	−504	0.66	198	Miss
2	1.03	618	−0.13	−39	Miss
3	0.92	552	0.06	18	Hit
4	−1.82	−1,092	−1.40	−420	Miss
5	−0.16	−96	0.23	69	Hit
6	−1.78	−1,068	1.33	399	Miss
7	2.04	1,224	0.69	207	Miss
8	1.08	648	−1.10	−330	Miss
9	−1.50	−900	−0.72	−216	Miss
10	−0.42	−252	−0.60	−180	Hit

[a] Total: 3 hits, 7 misses.

EXAMPLE 2.7 Lead-Time Demand

Lead-time demand may occur in an inventory system when the lead time is other than instantaneous. The lead time is the time from placement of an order until the order is received. In a realistic situation, lead time is a random variable. During the lead time, demands also occur at random. Lead-time demand is thus a random variable defined as the sum of the demands over the lead time, or $\sum_{i=0}^{T} D_i$, where i is the time period of the lead time, $i = 0, 1, 2, \ldots$; D_i is the demand during the ith time period; and T is the lead time. The distribution of lead-time demand is determined by simulating many cycles of lead time and building a histogram based on the results.

A firm sells bulk rolls of newsprint. The daily demand is given by the following probability distribution:

Daily Demand (Rolls)	3	4	5	6
Probability	0.20	0.35	0.30	0.15

The lead time is the number of days from placing an order until the firm receives the order from the supplier. In this instance, lead time is a random variable given by the following distribution:

Lead Time (Days)	1	2	3
Probability	0.36	0.42	0.22

Table 2.27 shows the random-digit assignment for demand, and Table 2.28 does the same for lead time. The incomplete simulation table is shown in Table 2.29. The random digits for the first cycle were 57. This generates a lead time of 2 days. Thus, two pairs of random digits must be generated for the

Table 2.27 Random-Digit Assignment for Demand

Daily Demand	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
3	0.20	0.20	01–20
4	0.35	0.55	21–55
5	0.30	0.85	56–85
6	0.15	1.00	86–00

daily demand. The first of these pairs is 87, which leads to a demand of 6. This is followed by a demand of 4. The lead-time demand for the first cycle is 10. After many cycles are simulated, a histogram is formulated. The histogram might appear as shown in Figure 2.9. This example illustrates how simulation can be used to study an unknown distribution by generating a random sample from the distribution. ▲

Table 2.28 Random-Digit Assignment for Lead Time

Lead Time (Days)	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
1	0.36	0.36	01–36
2	0.42	0.78	37–78
3	0.22	1.00	79–00

Table 2.29 Simulation Table for Lead-Time Demand

Cycle	*Random Digits for Lead Time*	*Lead Time (Days)*	*Random Digits for Demand*	*Demand*	*Lead-Time Demand*
1	57	2	87	6	
			34	4	10
2	33	1	82	5	5
3	93	3	28	4	
			19	3	
			63	5	12
4	55	2	91	6	
			26	4	10
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

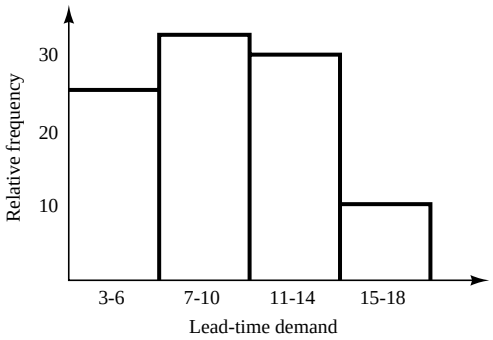


Figure 2.9 Histogram for lead-time demand.

2.4 Summary

This chapter introduced simulation concepts via examples in order to illustrate general areas of application and to motivate the remaining chapters. The next chapter gives a more systematic presentation of the basic concepts.

Ad hoc simulation tables were used in completing each example. Events in the tables were generated using uniformly distributed random numbers and, in one case, random normal numbers. The examples illustrate the need for determining the characteristics of the input data, generating random variables from the input models, and analyzing the resulting response. The queueing examples, especially the two-channel queue, illustrate some of the complex dependencies that can occur—in this example, between subsequent customers visiting the queue. Because of these complexities, the ad hoc simulation table approach fails, or becomes unbearably complex, even with relatively simple networks of queues. For this and other reasons, a more systematic methodology, such as the event-scheduling approach described in Chapter 3, is needed. These subjects are treated in more detail in the remaining chapters of the text.

Examples are drawn principally from queueing and inventory systems, because a large number of simulations concern problems in these areas. Additional examples are given in the areas of reliability, static simulation, and the generation of a random sample from an unknown distribution.

REFERENCES

HADLEY, G., and T. M. WHITIN [1963], *Analysis of Inventory Systems*, Prentice Hall, Englewood Cliffs, NJ.

HARRISON, J. M., and V. NGUYEN [1995], "Some Badly Behaved Closed Queueing Networks," *Proceedings of IMA Workshop on Stochastic Networks*, eds. F. Kelly and R. J. Williams, in press.

EXERCISES

If solving a problem manually with a simulation table, use different random digits taken from Table A.l for all problems referring to examples in the text. Many of the problems solved with a simulation table can be solved in a spreadsheet such as Excel. If using a spreadsheet, the reader is encouraged to investigate in the help system the existence and syntax for some common but useful functions. [For example, in Excel, use RAND() for random numbers between 0 and 1; IF() to return one of two values depending on whether a condition is true or false; MAX() and MIN() for maximums and minimums, etc.] Recall, also, that numerous hints for spreadsheet implementation were given in the text for the queueing examples.

1 In Example 2.1, let the arrival distribution be uniformly distributed between 1 and 10 minutes. Develop the simulation table and the analysis for 20 customers. What is the effect of changing the arrival-time distribution?

2 In Example 2.1, let the service distribution be changed to the following:

Service Time (Minutes)	1	2	3	4	5	6
Probability	0.05	0.10	0.20	0.30	0.25	0.10

Develop the simulation table and the analysis for 20 customers. What is the effect of changing the service-time distribution?

3 Perform the simulation in Example 2.1 for 20 more customers (customers 21 through 40). Compare the results of Example 2.1 to your results.

4 In Example 2.1, determine the time-weighted-average number of customers in the system and the time-weighted-average number of customers waiting. [*Hint:* Use Figure 2.6.]

5 In Example 2.2, change the arrival distribution of cars to the following:

Time between Arrivals (Minutes)	0	1	2	3	4
Probability	0.10	0.20	0.35	0.20	0.15

Develop the simulation and subsequent analysis for a period of 1 hour. What is the effect of changing the arrival distribution?

6 Again, with respect to Example 2.2, Able has a kneecap injury and cannot move as fast. Consequently, two things happen. Able's service distribution changes and Baker gets first shot at the customer if both carhops are idle. Able's new service distribution is as follows:

Service Time (Minutes)	3	4	5	6
Probability	0.30	0.30	0.25	0.15

(a) Develop a simulation and subsequent analysis for 30 service completions. What is the effect of Able's injury and the new rule?

(b) What would be the effect of adding a new employee who works at Baker's speed? The new employee would have all leftover work after Baker and Able.

7 Modify Example 2.2 so that Able has a probability of 0.45 of getting a customer in case both are idle. What is the effect of such a change?

8 Consider the following continuously operating job shop. Interarrival times of jobs are distributed as follows:

Time between Arrivals (Hours)	Probability
0	.23
1	.37
2	.28
3	.12

Processing times for jobs are normally distributed with mean 50 minutes and standard deviation 8 minutes. Construct a simulation table, and perform a simulation for 10 new customers. Assume that when the simulation begins there is one job being processed (scheduled to be completed in 25 minutes) and there is one job with a 50-minute processing time in the queue.

1. What was the average time in the queue for the 10 new jobs?
2. What was the average processing time of the 10 new jobs?
3. What was the maximum time in the system for the 10 new jobs?

9 Determine the optimal number of newspapers to be purchased daily in Example 2.3. Should the same random digits be used for each level of newspapers purchased by the seller? Why or why not? What effect does using new random numbers have?

10 A baker is trying to determine how many dozens of bagels to bake each day. The probability distribution of the number of bagel customers is as follows:

Number of Customers/Day	8	10	12	14
Probability	0.35	0.30	0.25	0.10

Customers order 1, 2, 3, or 4 dozen bagels according to the following probability distribution.

Number of Dozen Ordered/Customer	1	2	3	4
Probability	0.4	0.3	0.2	0.1

Bagels sell for $5.40 per dozen. They cost $3.80 per dozen to make. All bagels not sold at the end of the day are sold at half-price to a local grocery store. Based on 5 days of simulation, how many dozen (to the nearest 10 dozen) bagels should be baked each day?

11 Demand for widgets follows the probability distribution shown:

Daily Demand	0	1	2	3	4
Probability	0.33	0.25	0.20	0.12	0.10

Stock is examined every 7 days (the plant is in operation every day) and, if the stock level has reached 6 units, or less, an order for 10 widgets is placed. The lead time (days until delivery) is probabilistic and follows the following distribution:

Lead Time (Days)	1	2	3
Probability	0.3	0.5	0.2

When the simulation begins, it is the beginning of the week, 12 widgets are on hand, and no orders have been backordered. (Backordering is allowed.) Simulate 6 weeks of operation of this system. Analyze the system. Perform additional simulations to determine the effect on shortages if increases or decreases occur in (1) the review period, (2) the reorder quantity, and (3) the reorder point.

12 A plumbing supply firm is interested in the distribution of lead-time demand of industrial sinks. The probability distribution for daily demand is known and occurs as shown:

Daily Demand	0	1	2	3	4
Probability	0.18	0.39	0.29	0.09	0.05

The distribution of lead time has been reconstructed from past records as follows:

Lead Time (Days)	0	1	2	3	4	5
Probability	0.135	0.223	0.288	0.213	0.118	0.023

Develop the distribution of lead-time demand based on 20 cycles of lead time. Prepare a histogram of the distribution using intervals 0–2, 3–5, ... Then, prepare a histogram using intervals 0–1, 2–3, 4–5, ... Does changing the width of the interval have a pronounced effect on the form of the histogram of the lead-time demand distribution?

13 Develop and interpret flow diagrams analogous to Figures 2.2 and 2.3 for a queueing system with i channels.

14 Rework the simulation of the proposed method of Example 2.5 using new random digits. What is the effect on the total cost of generating a new set of events? When is it satisfactory to use the same random digits (and events) for competing proposals?

15 Smalltown Taxi operates one vehicle during the 9:00 A.M. to 5:00 P.M. period. Currently, consideration is being given to the addition of a second vehicle to the fleet. The demand for taxis follows the distribution shown:

Time between Calls (Minutes)	15	20	25	30	35
Probability	0.14	0.22	0.43	0.17	0.04

The distribution of time to complete a service is as follows:

Service Time (Minutes)	5	15	25	35	45
Probability	0.12	0.35	0.43	0.06	0.04

Simulate five individual days of operation of the current system and the system with an additional taxicab. Compare the two systems with respect to the waiting times of the customers and any other measures that might shed light on the situation.

16 Continue Example 2.6 for nine more runs and estimate how well the simulated bombers will do when they make their raid on the ammunition depot.

17 The random variables X, Y, and Z are distributed as follows:

$$X \sim N(\mu = 100, \sigma^2 = 100)$$
$$Y \sim N(\mu = 300, \sigma^2 = 225)$$
$$Z \sim N(\mu = 40, \sigma^2 = 64)$$

Simulate 50 values of the random variable

$$W = \frac{X + Y}{Z}$$

Prepare a histogram of the resulting values using class intervals of width equal to 3.

18 Given A, B, and C, three independent random variables: Variable A is normally distributed with $\mu = 100$ and $\sigma^2 = 400$. Variable B is discrete uniformly distributed with probability distribution given by $p(b) = 1/5$ with $b = 0, 1, 2, 3$ and 4. Variable C is distributed in accordance with the following table.

Value of C	Probability
10	.10
20	.25
30	.50
40	.15

Use simulation to estimate the mean of a new variable D, defined as

$$D = (A - 25B)/(2C)$$

Use a sample of size 10.

19 Lead time for a stock item is normally distributed with a mean of 7 days and a standard deviation of 2 days. Daily demand is distributed as follows:

Daily Demand	0	1	2	3	4
Probability	0.367	0.368	0.184	0.062	0.019

Determine the lead-time demand for 20 order cycles. (Round off lead time to the closest integer during the simulation and, if a negative value results, give it a lead time of zero.)

20 Consider Example 2.4.

(a) Extend the example for 15 more cycles and draw conclusions.

(b) Rework the example for 10 cycles with $M = 10$.

(c) Rework the example for 10 cycles with $N = 6$.

21 Estimate, by simulation, the average number of lost sales per week for an inventory system that functions as follows:

(a) Whenever the inventory level falls to or below 10 units, an order is placed. Only one order can be outstanding at a time.

(b) The size of each order is equal to $20 - I$, where I is the inventory level when the order is placed.

(c) If a demand occurs during a period when the inventory level is zero, the sale is lost.

(d) Daily demand is normally distributed with a mean of 5 units and a standard deviation of 1.5 units. (Round off demands to the closest integer during the simulation and, if a negative value results, give it a demand of zero.)

(e) Lead time is distributed uniformly between zero and 5 days, integers only.

(f) The simulation will start with 18 units in inventory.

(g) For simplicity, assume that all demands occur at 12 noon and that all orders are placed at the same time. Assume further that orders are received at 5:00 P.M., or after the demand which occurred on that day.

(h) Let the simulation run for 5 weeks.

22 An elevator in a manufacturing plant carries exactly 400 kilograms of material. There are three kinds of material, which arrive in boxes of known weight. These materials and their distributions of time between arrivals are as follows:

Material	Weight (kilograms)	Interarrival Time (Minutes)
A	200	5 ± 2(uniform)
B	100	6(constant)
C	50	$P(2) = 0.33$
		$P(3) = 0.67$

It takes the elevator 1 minute to go up to the second floor, 2 minutes to unload, and 1 minute to return to the first floor. The elevator does not leave the first floor unless it has a full load. Simulate 1 hour of operation of the system. What is the average transit time for a box of material A (time from its arrival until it is unloaded)? What is the average waiting time for a box of material B? How many boxes of material C made the trip in 1 hour?

23 The random variables X and Y are distributed as follows:

$$X \sim 10 \pm 10 \text{ (uniform)}$$

$$Y \sim 10 \pm 8 \text{ (uniform)}$$

(a) Simulate 200 values of the random variable

$$Z = XY$$

Prepare a histogram of the resulting values. What is the range of Z and what is its average value?

(b) Same as (a) except

$$Z = X/Y$$

24 Consider the assembly of two steel plates, each plate having a hole drilled in its center. The plates are to be joined by a pin. The plates are aligned for assembly relative to the bottom left corner $(0, 0)$. The hole placement is centered at $(3, 2)$ on each plate. The standard deviation in each direction is 0.0045. The hole diameter is normally distributed with a mean of 0.3 and a standard deviation of 0.005. The pin diameter is also distributed normally with a mean of 0.29 and a standard deviation of 0.004. What fraction of pins will go through the assembled plates? Base your answer on a simulation of 50 observations.

[*Hint:* Clearance $= \text{Min}(h_1, h_2) - [(x_1 - x_2)^2 + (y_1 - y_2)^2]^{.5} - p$, where

h_i = hole diameter, i = plate 1, 2

p = pin diameter

x_i = distance to center of plate hole, horizontal direction, $i = 1, 2$

y_i = distance to center of plate hole, vertical direction, $i = 1, 2$.]

25 In the exercise above, the pin will wobble if it is too loose. Wobbling occurs if $\text{Min}(h_1, h_2) - p \geq 0.006$. What fraction of the assemblies wobble? (Conditional probability, i.e., given that the pins go through.)

26 Three points are chosen at random on the circumference of a circle. Estimate the probability that they all lie on the same semicircle, using Monte Carlo sampling methods. Perform 5 replications.

27 Two theorems from statistics are as follows:

Theorem 1

Let $Z_1, Z_2, \ldots, Z_k$ be normally and independently distributed random variables with mean $\mu = 0$ and variance $\sigma^2 = 1$. Then the random variable

$$\chi^2 = Z_1^2 + Z_2^2 + \cdots + Z_k^2$$

is said to follow the chi-squared distribution with k degrees of freedom, abbreviated χ_k^2.

Theorem 2

Let $Z \sim N(0, 1)$ and V be a chi-square random variable with k degrees of freedom. If Z and V are independent, then the random variable

$$T = \frac{Z}{\sqrt{V/k}}$$

is said to follow the t distribution with k degrees of freedom, abbreviated t_k.

Generate a t-distributed random variate with 3 degrees of freedom. Use a spreadsheet and tabulate the generate values into a histogram; or, if manual, use the following random values in the order shown:

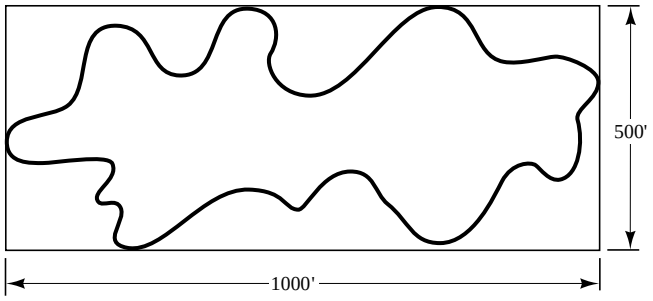
Random digits	Random normal numbers
6729	1.06
1837	−0.72
2572	0.28
8134	−0.18
5251	−0.63

28 A bank has one drive-in teller and room for one additional customer to wait. Customers arriving when the queue is full, park and go inside the bank to transact business. The time-between-arrivals and service-time distributions are given below.

Time between Arrivals (Minutes)	Probability	Service Time (Minutes)	Probability
0	0.09	1	0.20
1	0.17	2	0.40
2	0.27	3	0.28
3	0.20	4	0.12
4	0.15		
5	0.12		

Simulate the operation of the drive-in teller for 10 new customers. The first of the 10 new customers arrives at a time determined at random. Start the simulation with one customer being served, leaving at time 3, and one in the queue. How many customers went into the bank to transact business?

29 The number of acres comprised by Cedar Bog Lake is being estimated. The lake is shown in the accompanying sketch with the scale shown in feet. Use simulation to estimate the size of the lake. (One acre = 43,560 square feet).



3

General Principles

This chapter develops a common framework for the modeling of complex systems using discrete-event simulation. It covers the basic building blocks of all discrete-event simulation models: entities and attributes, activities and events. In discrete-event simulation, a system is modeled in terms of its state at each point in time; the entities that pass through the system and the entities that represent system resources; and the activities and events that cause system state to change. Discrete-event models are appropriate for those systems for which changes in system state occur only at discrete points in time.

The simulation languages and software (collectively called simulation packages) described in Chapter 4 are fundamentally packages for discrete-event simulation. A few of the packages also include the capability to model continuous variables in a purely continuous simulation or a mixed discrete-continuous model. The discussion in this chapter focuses on the discrete-event concepts and methodologies. The discussion in Chapter 4 focuses more on the capabilities of the individual packages and some of their higher-level constructs.

This chapter introduces and explains the fundamental concepts and methodologies underlying all discrete-event simulation packages. These concepts and methodologies are not tied to any particular package. Many of the packages use different terminology from that used here, and most have a number of higher-level constructs designed to make modeling simpler and more straightforward for their application domain. For example, while this chapter discusses the fundamental abstract concept of an entity, Chapter 4 discusses more concrete realizations of entities such as machines, conveyors, and vehicles that are built into some of the packages to facilitate modeling in the manufacturing, material handling or other domains.

Section 3.1 covers the general principles and concepts of discrete-event simulation, the event-scheduling/time-advance algorithm, and the three prevalent world views: eventscheduling, process interaction, and activity scanning. Section 3.2 introduces some of the notions of list processing, one of the more important methodologies used in discrete-event simulation software. Chapter 4 covers the implementation of the concepts in a number of the more widely used simulation packages.

3.1 Concepts in Discrete-Event Simulation

The concept of a system and a model of a system were discussed briefly in Chapter 1. This chapter deals exclusively with dynamic, stochastic systems (i.e., involving time and containing random elements) which change in a discrete manner. This section expands on these concepts and develops a framework for the development of a discrete-event model of a system. The major concepts are briefly defined and then illustrated by examples:

System A collection of entities (e.g., people and machines) that interact together over time to accomplish one or more goals.

Model An abstract representation of a system, usually containing structural, logical, or mathematical relationships which describe a system in terms of state, entities and their attributes, sets, processes, events, activities, and delays.

System state A collection of variables that contain all the information necessary to describe the system at any time.

Entity Any object or component in the system which requires explicit representation in the model (e.g., a server, a customer, a machine).

Attributes The properties of a given entity (e.g., the priority of a waiting customer, the routing of a job through a job shop).

List A collection of (permanently or temporarily) associated entities, ordered in some logical fashion (such as all customers currently in a waiting line, ordered by first come, first served, or by priority).

Event An instantaneous occurrence that changes the state of a system (such as an arrival of a new customer).

Event notice A record of an event to occur at the current or some future time, along with any associated data necessary to execute the event; at a minimum, the record includes the event type and the event time.

Event list A list of event notices for future events, ordered by time of occurrence; also known as the future event list (FEL).

Activity A duration of time of specified length (e.g., a service time or interarrival time), which is known when it begins (although it may be defined in terms of a statistical distribution).

Delay A duration of time of unspecified indefinite length, which is not known until it ends (e.g., a customer's delay in a last-in, first-out waiting line which, when it begins, depends on future arrivals).

Clock A variable representing simulated time, called CLOCK in the examples to follow.

Different simulation packages use different terminology for the same or similar concepts. For example, lists are sometimes called sets, queues, or chains. Sets or lists are used to hold entities as well as event notices. The entities on a list are always ordered by some rule, such as first-in first-out or last-in first-out, or ranked by some entity attribute, such as priority or due date. The future event list is always ranked by the event time recorded in the event notice. Section 3.2 discusses a number of methods for handling lists and introduces some of the methodologies for efficient processing of ordered sets or lists.

An activity typically represents a service time, an interarrival time, or any other processing time whose duration has been characterized and defined by the modeler. An activity's duration may be specified in a number of ways:

1. Deterministic—for example, always exactly 5 minutes;
2. Statistical—for example, as a random draw from among 2, 5, 7 with equal probabilities;
3. A function depending on system variables and/or entity attributes—for example, loading time for an iron ore ship as a function of the ship's allowed cargo weight and the loading rate in tons per hour.

However it is characterized, the duration of an activity is computable from its specification at the instant it begins. Its duration is not affected by the occurrence of other events (unless, as is allowed by some simulation packages, the model contains logic to cancel an event). To keep track of activities and their expected completion time, at the simulated instant that an activity duration begins, an event notice is created having an event time equal to the activity's completion time. For example, if the current simulated time is CLOCK $= 100$ minutes and an inspection time of exactly 5 minutes is just beginning, then an event notice is created that specifies the type of event (an end of inspection event) and the event time ($100 + 5 = 105$ minutes).

In contrast to an activity, a delay's duration is not specified by the modeler ahead of time, but rather is determined by system conditions. Quite often, a delay's duration is measured and is one of the desired outputs of a model run. Typically, a delay ends when some set of logical conditions becomes true or one or more other events occur. For example, a customer's delay in a waiting line may be dependent on the number and duration of service of other customers ahead in line as well as the availability of servers and equipment.

A delay is sometimes called a *conditional wait*, while an activity is called an *unconditional wait*. The completion of an activity is an event, often called a *primary event*, that is managed by placing an event notice on the FEL. In contrast, delays are managed by placing the associated entity on another list, perhaps representing a waiting line, until such time as system conditions permit the processing of the entity. The completion of a delay is sometimes called a

conditional or secondary event, but such events are not represented by event notices, nor do they appear on the FEL.

The systems considered here are dynamic—that is, changing over time. Therefore, system state, entity attributes and the number of active entities, the contents of sets, and the activities and delays currently in progress are all functions of time and are constantly changing over time. Time itself is represented by a variable called CLOCK.

EXAMPLE 3.1 (Able and Baker, Revisited)

Consider the Able–Baker carhop system of Example 2.2. A discrete-event model has the following components:

System state

$L_Q(t)$, the number of cars waiting to be served at time t

$L_A(t)$, 0 or 1 to indicate Able being idle or busy at time t

$L_B(t)$, 0 or 1 to indicate Baker being idle or busy at time t

Entities Neither the customers (i.e., cars) nor the servers need to be explicitly represented, except in terms of the state variables, unless certain customer averages are desired (compare Examples 3.4 and 3.5)

Events

Arrival event

Service completion by Able

Service completion by Baker

Activities

Interarrival time, defined in Table 2.11

Service time by Able, defined in Table 2.12

Service time by Baker, defined in Table 2.13

Delay A customer's wait in queue until Able or Baker becomes free ▲

The definition of the model components provides a static description of the model. In addition, a description of the dynamic relationships and interactions between the components is also needed. Some questions that need answers include:

1. How does each event affect system state, entity attributes, and set contents?
2. How are activities defined (i.e., deterministic, probabilistic, or some other mathematical equation)? What event marks the beginning or end of each activity? Can the activity begin regardless of system state, or is its beginning conditioned on the system being in a certain state? (For example, a machining "activity" cannot begin unless the machine is idle, not broken, and not in maintenance.)
3. Which events trigger the beginning (and end) of each type of delay? Under what conditions does a delay begin, or end?
4. What is the system state at time 0? What events should be generated at time 0 to "prime" the model—that is, to get the simulation started?

A discrete-event simulation is the modeling over time of a system all of whose state changes occur at discrete points in time—those points when an event occurs. A discrete-event simulation (hereafter called a simulation) proceeds by producing a sequence of system snapshots (or system images) which represent the evolution of the system through time. A given snapshot at a given time (CLOCK $= t$) includes not only the system state at time t, but also a list (the FEL) of all activities currently in progress and when each such activity will end, the status of all entities and current membership of all sets, plus the current values of cumulative statistics and counters that will be used to calculate summary statistics at the end of the simulation. A prototype system snapshot is shown in Figure 3.1. (Not all models will contain every element exhibited in Figure 3.1. Further illustrations are provided in the examples in this chapter.)

3.1.1 The Event-Scheduling/Time-Advance Algorithm

The mechanism for advancing simulation time and guaranteeing that all events occur in correct chronological order is based on the future event list (FEL). This list contains all event notices for events that have been scheduled to occur at a future time. Scheduling a future event means that at the instant an activity begins, its duration is computed or drawn as a sample from a statistical distribution and the end-activity event, together with its event time, is placed on the future event list. In the real world, most future events are not scheduled but merely happen—such as random breakdowns or random arrivals. In the model, such random events are represented by the end of some activity, which in turn is represented by a statistical distribution.

At any given time t, the FEL contains all previously scheduled future events and their associated event times (called $t_1, t_2, \ldots$ in Figure 3.1). The FEL is ordered by event time, meaning that the events are arranged chronologically; that is, the event times satisfy

$$t < t_1 \leq t_2 \leq t_3 \leq \cdots \leq t_n$$

Time t is the value of CLOCK, the current value of simulated time. The event associated with time t_1 is called the imminent event; that is, it is the next event that will occur. After the system snapshot at simulation time CLOCK $= t$ has been updated, the CLOCK is advanced to simulation time CLOCK $= t_1$, and the imminent event notice is removed from the FEL and the event executed. Execution of the imminent event means that a new system snapshot for time t_1 is created based on the old snapshot at time t and the nature of the imminent event. At time t_1, new future events may or may not be generated, but if any are, they are scheduled by creating event notices and putting them in their proper position on the FEL. After the new system snapshot for time t_1 has been updated, the clock is advanced to the time of the new imminent event and that event is executed. This process repeats until the simulation is

Clock	*System State*	*Entities and Attributes*	*Set 1*	*Set 2*	...	*Future Event List, FEL*	*Cumulative Statistics and Counters*
t	$(x, y, z, \ldots)$					$(3, t_1)$– Type 3 event to occur at time t_1 $(1, t_2)$– Type 1 event to occur at time t_2	
	.					. .	
	.					. .	
	.					. .	

Figure 3.1 Prototype system snapshot at simulation time t.

over. The sequence of actions which a simulator (or simulation language) must perform to advance the clock and build a new system snapshot is called the *event-scheduling/time-advance algorithm*, whose steps are listed in Figure 3.2 (and explained below).

Old system snapshot at time t

CLOCK	*System State*	...	*Future Event List*	...
t	(5, 1, 6)		$(3, t_1)$– Type 3 event to occur at time t_1 $(1, t_2)$– Type 1 event to occur at time t_2 $(1, t_3)$– Type 1 event to occur at time t_3 . . . . $(2, t_n)$– Type 2 event to occur at time t_n	

Event-scheduling/time-advance algorithm

Step 1. Remove the event notice for the imminent event (event 3, time t_1) from FEL

Step 2. Advance CLOCK to imminent event time (i.e., advance CLOCK from t to t_1).

Step 3. Execute imminent event: update system state, change entity attributes, and set membership as needed.

Step 4. Generate future events (if necessary) and place their event notices on FEL ranked by event time. (*Example:* Event 4 to occur at time t^*, where $t_2 < t^* < t_3$.)

Step 5. Update cumulative statistics and counters.

New system snapshot at time t_1

CLOCK	*System State*	...	*Future Event List*	...
t_1	(5, 1, 5)		$(1, t_2)$– Type 1 event to occur at time t_2 $(4, t^*)$– Type 4 event to occur at time t^* $(1, t_3)$– Type 1 event to occur at time t_3 . . . . $(2, t_n)$– Type 2 event to occur at time t_n	

Figure 3.2 Advancing simulation time and updating system image.

The length and contents of the FEL are constantly changing as the simulation progresses, and thus its efficient management in a computerized simulation will have a major impact on the efficiency of the computer program representing the model. The management of a list is called *list processing*. The major list-processing operations performed on a FEL are removal of the imminent event, addition of a new event to the list, and occasionally removal of some event (called cancellation of an event). As the imminent event is usually at the top of the list, its removal is as efficient as possible. Addition of a new event (and cancellation of an old event) requires a search of the list. The efficiency of this search depends on the logical organization of the list and on how the search is conducted. In addition to the FEL, all the sets in a model are maintained in some logical order, and the operations of addition and removal of entities from the set also require efficient list-processing techniques. A brief introduction to list processing in simulation is given in Section 3.2.

The removal and addition of events from the FEL is illustrated in Figure 3.2. Event 3 with event time t_1 represents, say, a service completion event at server 3. Since it is the imminent event at time t, it is removed from the FEL in step 1 (Figure 3.2) of the event-scheduling/time-advance algorithm. When event 4 (say, an arrival event) with event time t^* is generated at step 4, one possible way to determine its correct position on the FEL is to conduct a top-down search:

If $t^* < t_2$,	place event 4 at the top of the FEL.
If $t_2 \le t^* < t_3$,	place event 4 second on the list.
If $t_3 \le t^* < t_4$,	place event 4 third on the list.
$\vdots$	
If $t_n \le t^*$,	place event 4 last on the list.

(In Figure 3.2, it was assumed that t^* was between t_2 and t_3.) Another way is to conduct a bottom-up search. The least efficient way to maintain the FEL is to leave it as an unordered list (additions placed arbitrarily at the top or bottom), which would require at step 1 of Figure 3.2 a complete search of the list for the imminent event before each clock advance. (The imminent event is the event on the FEL with the lowest event time.)

The system snapshot at time 0 is defined by the initial conditions and the generation of the so-called exogenous events. The specified initial conditions define the system state at time 0. For example, in Figure 3.2, if $t = 0$, then the state (5, 1, 6) might represent the initial number of customers at three different points in the system. An exogenous event is a happening "outside the system" which impinges on the system. An important example is an arrival to a queueing system. At time 0, the first arrival event is generated and is scheduled on the FEL (meaning its event notice is placed on the FEL). The interarrival time is an example of an activity. When the clock eventually is advanced to the time of this first arrival, a second arrival event is generated. First, an interarrival time is generated, a^*; it is added to the current time, CLOCK $= t$; the resulting (future) event time, $t + a^* = t^*$, is used to position the new arrival event

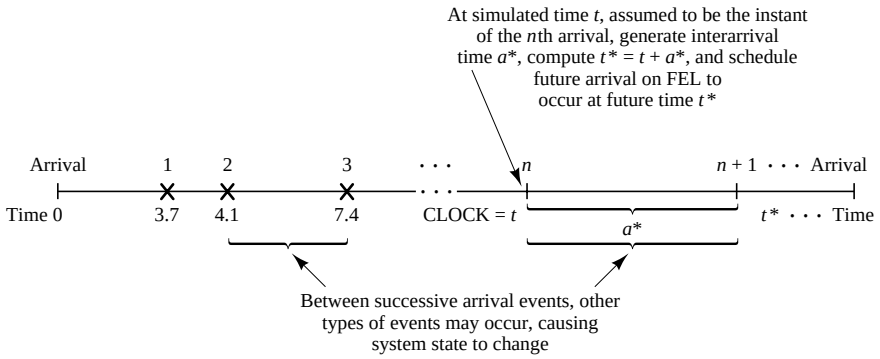


Figure 3.3 Generation of an external arrival stream by bootstrapping.

notice on the FEL. This method of generating an external arrival stream, called *bootstrapping*, provides one example of how future events are generated in step 4 of the event-scheduling/time-advance algorithm. Bootstrapping is illustrated in Figure 3.3. The first three interarrival times generated are 3.7, 0.4, and 3.3 time units. The end of an interarrival interval is an example of a primary event.

A second example of how future events are generated (step 4 of Figure 3.2) is provided by a service-completion event in a queueing simulation. When one customer completes service, at current time CLOCK $= t$, if the next customer is present, then a new service time, s^*, will be generated for the next customer. The next service-completion event will be scheduled to occur at future time $t^* = t + s^*$ by placing onto the FEL a new event notice of type service completion with event time t^*. In addition, a service-completion event will be generated and scheduled at the time of an arrival event, provided that, upon arrival, there is at least one idle server in the server group. A service time is an example of an activity. Beginning service is a conditional event, because its occurrence is triggered only on the condition that a customer is present and a server is free. Service completion is an example of a primary event. Note that a conditional event, such as beginning service, is triggered by a primary event occurring and certain conditions prevailing in the system. Only primary events appear on the FEL.

A third important example is the alternate generation of runtimes and downtimes for a machine subject to breakdowns. At time 0, the first runtime will be generated and an end-of-runtime event scheduled. Whenever an end-of-runtime event occurs, a downtime will be generated and an end-of-downtime event scheduled on the FEL. When the CLOCK is eventually advanced to the time of this end-of-downtime event, a runtime is generated and an end-of-runtime event scheduled on the FEL. In this way, runtimes and downtimes continually alternate throughout the simulation. A runtime and a downtime are examples of activities, and end of runtime and end of downtime are primary events.

Every simulation must have a stopping event, here called E, which defines how long the simulation will run. There are generally two ways to stop a simulation:

1. At time 0, schedule a stop simulation event at a specified future time T_E. Thus, before simulating, it is known that the simulation will run over the time interval $[0, T_E]$. *Example:* Simulate a job shop for $T_E = 40$ hours.
2. Run length T_E is determined by the simulation itself. Generally, T_E is the time of occurrence of some specified event E. *Examples:* T_E is the time of the 100th service completion at a certain service center. T_E is the time of breakdown of a complex system. T_E is the time of disengagement or total kill (whichever occurs first) in a combat simulation. T_E is the time at which a distribution center ships the last carton in a day's orders.

In case 2, T_E is not known ahead of time. Indeed, it may be one of the statistics of primary interest to be produced by the simulation.

3.1.2 World Views

When using a simulation package or even when doing a manual simulation, a modeler adopts a world view or orientation for developing a model. Those most prevalent are the event-scheduling world view, as discussed in the previous section, the process-interaction world view, and the activity-scanning world view. Even if a particular package does not directly support one or more of the world views, understanding the different approaches may suggest alternative ways to model a given system.

To summarize the previous discussion, when using the event-scheduling approach, a simulation analyst concentrates on events and their effect on system state. This world view will be illustrated by the manual simulations of Section 3.1.3 and the C++ simulation in Chapter 4.

When using a package that supports the process-interaction approach, a simulation analyst thinks in terms of processes. The analyst defines the simulation model in terms of entities or objects and their life cycle as they flow through the system, demanding resources and queueing to wait for resources. More precisely, a process is the life cycle of one entity. This life cycle consists of various events and activities. Some activities may require the use of one or more resources whose capacities are limited. These and other constraints cause processes to interact, the simplest example being an entity forced to wait in a queue (on a list) because the resource it needs is busy with another entity. The process-interaction approach is popular because of its intuitive appeal, and because the simulation packages that implement it allow an analyst to describe the process flow in terms of high-level block or network constructs, while the interaction among processes is handled automatically.

In more precise terms, a process is a time-sequenced list of events, activities, and delays, including demands for resources, that define the life cycle

of one entity as it moves through a system. An example of a "customer process" is shown in Figure 3.4. In this figure, we see the interaction between two customer processes as customer $n + 1$ is delayed until the previous customer's "end-service event" occurs. Usually many processes are simultaneously active in a model, and the interaction among processes may be quite complex.

Underlying the implementation of the process-interaction approach in a simulation package, but usually hidden from a modeler's view, events are being scheduled on a future event list and entities are being placed onto lists whenever they face delays, causing one process to temporarily suspend its execution while other processes proceed. It is important that the modeler have a basic understanding of the concepts and, for the simulation package being used, a detailed understanding of the built-in but hidden rules of operation. Schriber and Brunner [1998] provide provide understanding in this area.

Both the event-scheduling and the process-interaction approaches use a variable time advance; that is, when all events and system state changes have occurred at one instant of simulated time, the simulation clock is advanced to the time of the next imminent event on the FEL. The activity-scanning approach, in contrast, uses a fixed time increment and a rule-based approach to decide whether any activities can begin at each point in simulated time.

With the activity-scanning approach, a modeler concentrates on the activities of a model and those conditions, simple or complex, that allow an activity to begin. At each clock advance, the conditions for each activity are checked and, if the conditions are true, then the corresponding activity begins. Proponents claim that the activity-scanning approach is simple in concept and leads to modular models that are more maintainable and easily understood and modified by other analysts at later times. They admit, however, that the repeated scanning to determine whether an activity can begin results in slow runtime on computers. Thus, the pure activity-scanning approach has been modified (and made conceptually somewhat more complex) by what is called the three-phase approach, which combines some of the features of event scheduling with activity scanning to allow for variable time advance and the avoidance of scanning when it is not necessary, but keeping the main advantages of the activity-scanning approach.

In the three-phase approach, events are considered to be activities of duration-zero time units. With this definition, activities are divided into two categories, called B and C.

B activities activities bound to occur; all primary events and unconditional activities.

C activities activities or events that are conditional upon certain conditions being true.

The B-type activities and events can be scheduled ahead of time, just as in the event-scheduling approach. This allows variable time advance. The FEL contains only B-type events. Scanning to check if any C-type activities can begin

or C-type events occur happens only at the end of each time advance, after all B-type events have completed. In summary, with the three-phase approach, the simulation proceeds with repeated execution of the three phases until it is completed:

Phase A Remove the imminent event from the FEL and advance the clock to its event time. Remove any other events from the FEL that have the same event time.

Phase B Execute all B-type events that were removed from the FEL. (This may free a number of resources or otherwise change system state.)

Phase C Scan the conditions that trigger each C-type activity and activate any whose conditions are met. Rescan until no additional C-type activities can begin or events occur.

The three-phase approach improves the execution efficiency of the activity-scanning method. In addition, proponents claim that the activity-scanning and three-phase approaches are particularly good at handling complex resource problems, in which various combinations of resources are needed to accomplish different tasks. These approaches guarantee that resources being freed at a given simulated time will all be freed before any available resources are reallocated to new tasks.

EXAMPLE 3.2 (Able and Baker, Back Again)

The events and activities were identified in Example 3.1. Using the three-phase approach, the conditions for beginning each activity in Phase C are:

Activity	Condition
Service time by Able	A customer is in queue and Able is idle
Service time by Baker	A customer is in queue, Baker is idle, and Able is busy

Using the process-interaction approach, we view the model from the viewpoint of a customer and its "life cycle." Considering a life cycle beginning upon arrival, a customer process is pictured in Figure 3.4. ▲

In summary, as will be illustrated in Chapter 4, the process-interaction approach has been adopted by the simulation packages most popular in the United States. On the other hand, a number of activity scanning-based packages are popular in the United Kingdom and Europe. Some of the packages allow portions of a model to be event-scheduling based, if that orientation is convenient, mixed with the process-interaction approach. Finally, some of the packages are based on a flow chart, block diagram, or network structure, which upon closer examination turns out to be a specific implementation of the process-interaction concept.

3.1.3 Manual Simulation Using Event Scheduling

In conducting an event-scheduling simulation, a simulation table is used to record the successive system snapshots as time advances.

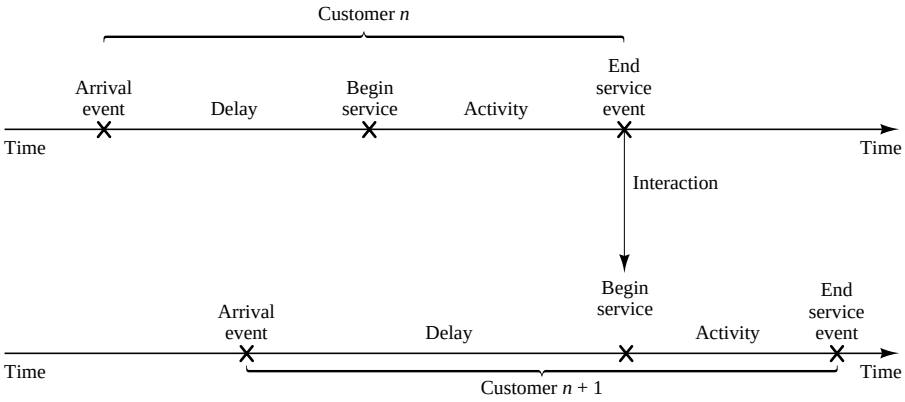


Figure 3.4 Two interacting customer processes in a single-server queue.

EXAMPLE 3.3 (Single-Channel Queue)

Reconsider the grocery store with one checkout counter that was simulated in Example 2.1 by an ad hoc method. The system consists of those customers in the waiting line plus the one (if any) checking out. The model has the following components:

System state $(LQ(t), LS(t))$, where $LQ(t)$ is the number of customers in the waiting line, and $LS(t)$ is the number being served (0 or 1) at time t.

Entities The server and customers are not explicitly modeled, except in terms of the state variables above.

Events

Arrival (A)

Departure (D)

Stopping event (E), scheduled to occur at time 60.

Event notices

(A, t), representing an arrival event to occur at future time t

(D, t), representing a customer departure at future time t

$(E, 60)$, representing the simulation-stop event at future time 60.

Activities

Interarrival time, defined in Table 2.6

Service time, defined in Table 2.7

Delay Customer time spent in waiting line.

The event notices are written as (event type, event time). In this model, the FEL will always contain either two or three event notices. The effect of the arrival and departure events was first shown in Figures 2.2 and 2.3 and is shown in more detail in Figures 3.5 and 3.6.

The simulation table for the checkout counter is given in Table 3.1. The reader should cover all system snapshots except one, starting with the first, and attempt to construct the next snapshot from the previous one and the event

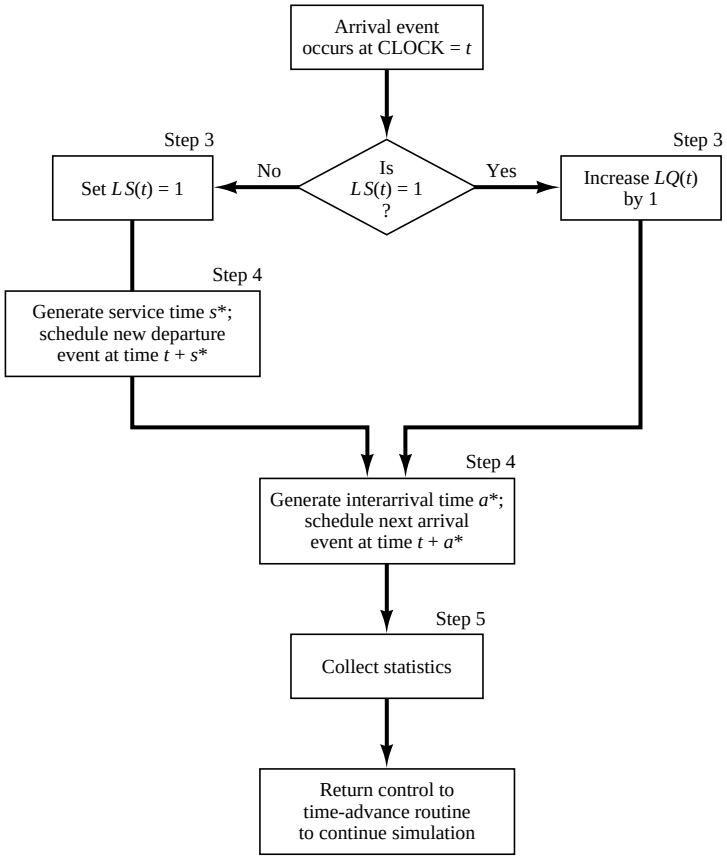


Figure 3.5 Execution of the arrival event.

logic in Figures 3.5 and 3.6. The interarrival times and service times will be identical to those used in Table 2.10, namely:

Interarrival Times	8	6	1	8	3	8	···
Service Times	4	1	4	3	2	4	···

Initial conditions are that the first customer arrives at time 0 and begins service. This is reflected in Table 3.1 by the system snapshot at time zero (CLOCK $= 0$), with $LQ(0) = 0, LS(0) = 1$, and both a departure event and arrival event on the FEL. Also, the simulation is scheduled to stop at time 60. Only two statistics, server utilization and maximum queue length, will be collected. Server utilization is defined by total server busy time (B) divided by total time (T_E). Total busy time, B, and maximum queue length, MQ, will be accumulated as the simulation progresses. A column headed "comments" is included to aid the reader. (a^* and s^* are the generated interarrival and service times, respectively.)

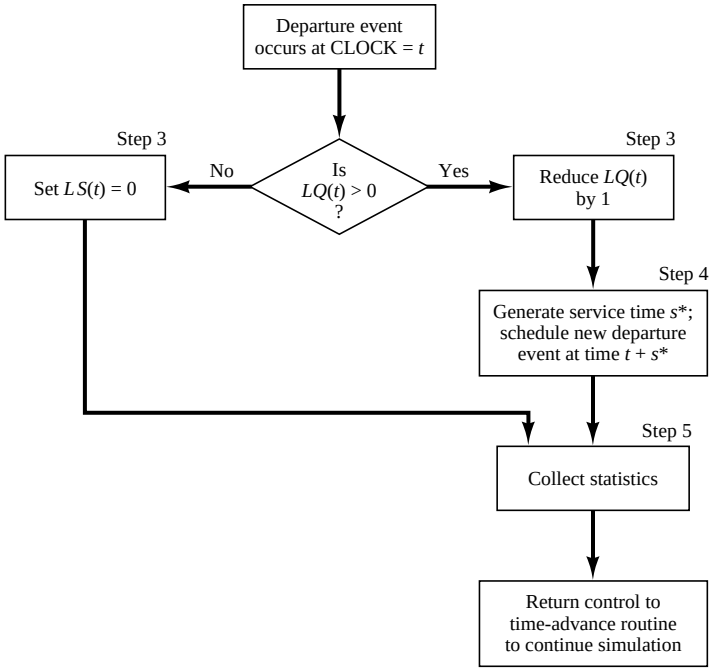


Figure 3.6 Execution of the departure event.

As soon as the system snapshot at time CLOCK = 0 is complete, the simulation begins. At time 0, the imminent event is $(D, 4)$. The CLOCK is advanced to time 4, and $(D, 4)$ is removed from the FEL. Since $LS(t) = 1$ for $0 \leq t \leq 4$ (i.e., the server was busy for 4 minutes), the cumulative busy time is increased from $B = 0$ to $B = 4$. By the event logic in Figure 3.6, set $LS(4) = 0$ (the server becomes idle). The FEL is left with only two future events, $(A, 8)$ and $(E, 0)$. The simulation CLOCK is next advanced to time 8 and an arrival event executed. The interpretation of the remainder of Table 3.1 is left to the reader.

The simulation in Table 3.1 covers the time interval $[0, 21]$. At simulated time 21, the system is empty, but the next arrival will occur at future time 23. The server was busy for 12 of the 21 time units simulated, and the maximum queue length was one. This simulation is, of course, too short to allow us to draw any reliable conclusions. Exercise 1 asks the reader to continue the simulation and compare the results to those in Example 2.1. Note that the simulation table gives the system state at all times, not just the listed times. For example, from time 15 to time 18, there is one customer in service and one in the waiting line.

Table 3.1 Simulation Table for Checkout Counter (Example 3.3)

	System State				*Cumulative Statistics*	
Clock	$LQ(t)$	$LS(t)$	*Future Event List*	*Comment*	B	MQ
0	0	1	$(D, 4)$ $(A, 8)$ $(E, 60)$	First A occurs $(a^* = 8)$ Schedule next A $(s^* = 4)$ Schedule first D	0	0
4	0	0	$(A, 8)$ $(E, 60)$	First D occurs: $(D, 4)$	4	0
8	0	1	$(D, 9)$ $(A, 14)$ $(E, 60)$	Second A occurs: $(A, 8)$ $(a^* = 6)$ Schedule next A $(s^* = 1)$ Schedule next D	4	0
9	0	0	$(A, 14)$ $(E, 60)$	Second D occurs: $(D, 9)$	5	0
14	0	1	$(A, 15)$ $(D, 18)$ $(E, 60)$	Third A occurs: $(A, 14)$ $(s^* = 4)$ Schedule next D	5	0
15	1	1	$(D, 18)$ $(A, 23)$ $(E, 60)$	Fourth A occurs: $(A, 15)$ (Customer delayed)	6	1
18	0	1	$(D, 21)$ $(A, 23)$ $(E, 60)$	Third D occurs: $(D, 18)$ $(s^* = 3)$ Schedule next D	9	1
21	0	0	$(A, 23)$ $(E, 60)$	Fourth D occurs: $(D, 21)$	12	1

When an event-scheduling algorithm is computerized, only one snapshot (the current one or partially updated one) is kept in computer memory. With the idea of implementing event scheduling in C++ or some other general-purpose language, the following rule should be followed. A new snapshot can be derived only from the previous snapshot, newly generated random variables, and the event logic (Figures 3.5 and 3.6). Past snapshots should be ignored when advancing the clock. The current snapshot must contain all information necessary to continue the simulation. ▲

EXAMPLE 3.4 (The Checkout-Counter Simulation, Continued)

Suppose that in the simulation of the checkout counter in Example 3.3 the simulation analyst desires to estimate mean response time and mean proportion of customers who spend 4 or more minutes in the system. A response time is the length of time a customer spends in the system. In order to estimate these customer averages, it is necessary to expand the model in Example 3.3 to explicitly represent the individual customers. In addition, to be able to compute an individual customer's response time when that customer departs, it will be necessary to know that customer's arrival time. Therefore, a customer entity with arrival time as an attribute will be added to the list of model components in Example 3.3. These customer entities will be stored in a list to be called "CHECKOUT LINE"; they will be called $C1, C2, C3, \ldots$. Finally, the event notices on the FEL will be expanded to indicate which customer is affected.

For example, $(D, 4, C1)$ means that customer $C1$ will depart at time 4. The additional model components are listed below:

Entities (Ci, t), representing customer Ci who arrived at time t

Event notices

(A, t, Ci), the arrival of customer Ci at future time t

(D, t, Cj), the departure of customer Cj at future time t

Set "CHECKOUT LINE," the set of all customers currently at the checkout counter (being served or waiting to be served), ordered by time of arrival

Three new cumulative statistics will be collected: S, the sum of customer response times for all customers who have departed by the current time; F, the total number of customers who spend 4 or more minutes at the checkout counter; and N_D the total number of departures up to the current simulation time. These three cumulative statistics will be updated whenever the departure event occurs; the logic for collecting these statistics would be incorporated into step 5 of the departure event in Figure 3.6.

The simulation table for Example 3.4 is shown in Table 3.2. The same data for interarrival and service times will be used again, so that Table 3.2 essentially repeats Table 3.1, except that the new components are included (and the comment column has been deleted). These new components are needed for the computation of the cumulative statistics S, F, and N_D. For example, at time 4 a departure event occurs for customer $C1$. The customer entity $C1$ is removed from the list called "CHECKOUT LINE"; the attribute "time of arrival" is noted to be 0, so the response time for this customer was 4 minutes. Hence, S is incremented by 4 minutes, and F and N_D are incremented by one customer. Similarly, at time 21 when the departure event $(D, 21, C4)$ is being executed, the response time for customer $C4$ is computed by

$$\begin{aligned} \text{Response time} &= \text{CLOCK TIME} - \text{attribute "time of arrival"} \\ &= \quad 21 \quad - \quad 15 \\ &= 6 \text{ minutes} \end{aligned}$$

Then S is incremented by 6 minutes, and F and N_D by one customer.

For a simulation run length of 21 minutes, the average response time was $S/N_D = 15/4 = 3.75$ minutes, and the observed proportion of customers who spent 4 or more minutes in the system was $F/N_D = 0.75$. Again, this simulation was far too short to allow us to regard these estimates with any degree of accuracy. The purpose of Example 3.4, however, was to illustrate the notion that in many simulation models the information desired from the simulation (such as the statistics S/N_D and F/N_D) determines to some extent the structure of the model. ▲

Table 3.2 Simulation Table for Example 3.4

	System State		*List*	*Future Event*	*Cumulative Statistics*		
Clock	$LQ(t)$	$LS(t)$	*"CHECKOUT LINE"*	*List*	S	N_D	F
0	0	1	$(C1, 0)$	$(D, 4, C1)$ $(A, 8, C2)$ $(E, 60)$	0	0	0
4	0	0		$(A, 8, C2)$ $(E, 60)$	4	1	1
8	0	1	$(C2, 8)$	$(D, 9, C2)$ $(A, 14, C3)$ $(E, 60)$	4	1	1
9	0	0		$(A, 14, C3)$ $(E, 60)$	5	2	1
14	0	1	$(C3, 14)$	$(A, 15, C4)$ $(D, 18, C3)$ $(E, 60)$	5	2	1
15	1	1	$(C3, 14)$ $(C4, 15)$	$(D, 18, C3)$ $(A, 23, C5)$ $(E, 60)$	5	2	1
18	0	1	$(C4, 15)$	$(D, 21, C4)$ $(A, 23, C5)$ $(E, 60)$	9	3	2
21	0	0		$(A, 23, C5)$ $(E, 60)$	15	4	3

EXAMPLE 3.5 (The Dump Truck Problem)

Six dump trucks are used to haul coal from the entrance of a small mine to the railroad. Figure 3.7 provides a schematic of the dump truck operation. Each truck is loaded by one of two loaders. After loading, a truck immediately moves to the scale, to be weighed as soon as possible. Both the loaders and the scale have a first-come first-served waiting line (or queue) for trucks. Travel time from a loader to the scale is considered negligible. After being weighed, a truck begins a travel time (during which time the truck unloads) and then afterward returns to the loader queue. The distributions of loading time, weighing time, and travel time are given in Tables 3.3, 3.4, and 3.5, respectively, together with the random digit assignment for generating these variables using random digits from Table A.1. The purpose of the simulation is to estimate the loader and scale utilizations (percentage of time busy). The model has the following components:

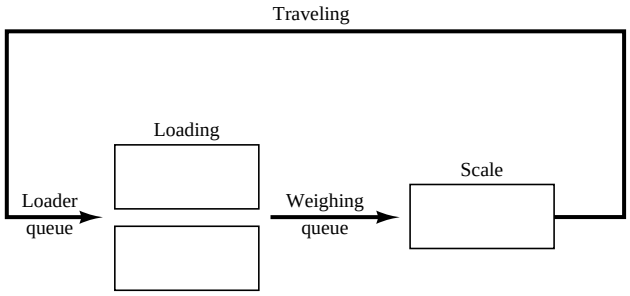


Figure 3.7 Dump truck problem.

Table 3.3 Distribution of Loading Time for the Dump Trucks

Loading Time	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
5	0.30	0.30	1–3
10	0.50	0.80	4–8
15	0.20	1.00	9–0

System state

$[LQ(t), L(t), WQ(t), W(t)]$, where

$LQ(t)$ = number of trucks in loader queue

$L(t)$ = number of trucks (0, 1, or 2) being loaded

$WQ(t)$ = number of trucks in weigh queue

$W(t)$ = number of trucks (0 or 1) being weighed, all at simulation time t

Event notices

(ALQ, t, DTi), dump truck i arrives at loader queue (ALQ) at time t

(EL, t, DTi), dump truck i ends loading (EL) at time t

(EW, t, DTi), dump truck i ends weighing (EW) at time t

Entities The six dump trucks $(DT1, \ldots, DT6)$

Lists

Loader queue, all trucks waiting to begin loading, ordered on a first come, first served basis

Weigh queue, all trucks waiting to be weighed, ordered on a first come, first served basis

Activities Loading time, weighing time, and travel time

Delays Delay at loader queue, and delay at scale

Table 3.4 Distribution of Weighing Time for the Dump Trucks

Weighing Time	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
12	0.70	0.70	1–7
16	0.30	1.00	8–0

Table 3.5 Distribution of Travel Time for the Dump Trucks

Travel Time	*Probability*	*Cumulative Probability*	*Random-Digit Assignment*
40	0.40	0.40	1–4
60	0.30	0.70	5–7
80	0.20	0.90	8–9
100	0.10	1.00	0

The simulation table is given in Table 3.6. It has been assumed that five of the trucks are at the loaders and one is at the scale at time 0. The activity times are taken from the following list as needed:

Loading Time	10	5	5	10	15	10	10
Weighing Time	12	12	12	16	12	16	
Travel Time	60	100	40	40	80		

When an end-loading (EL) event occurs, say for truck j at time t, other events may be triggered. If the scale is idle [$W(t) = 0$], truck j begins weighing and an end-weighing event (EW) is scheduled on the FEL; otherwise, truck j joins the weigh queue. If at this time there is another truck waiting for a loader, it will be removed from the loader queue and will begin loading by the scheduling of an end-loading event (EL) on the FEL. This logic for the occurrence of the end-loading event, as well as the appropriate logic for the other two events, should be incorporated into an event diagram as in Figures 3.5 and 3.6 of Example 3.3. The construction of these event logic diagrams is left as an exercise for the reader (Exercise 2).

As an aid to the reader, in Table 3.6 whenever a new event is scheduled, its event time is written as "$t+$ (activity time)." For example, at time 0 the imminent event is an EL event with event time 5. The clock is advanced to time $t = 5$, dump truck 3 joins the weigh queue (since the scale is occupied), and truck 4 begins to load. Thus, an EL event is scheduled for truck 4 at future time 10, computed by (present time) + (loading time) $= 5 + 5 = 10$.

In order to estimate the loader and scale utilizations, two cumulative statistics are maintained:

B_L = total busy time of both loaders from time 0 to time t

B_S = total busy time of the scale from time 0 to time t

Since both loaders are busy from time 0 to time 20, $B_L = 40$ at time $t = 20$. But from time 20 to time 24, only one loader is busy; thus, B_L increases by only 4 minutes over the time interval [20, 24]. Similarly, from time 25 to time 36, both loaders are idle ($L(25) = 0$), so B_L does not change. For the relatively

Table 3.6 Simulation Table for Dump Truck Operation (Example 3.5)

	System State				*Lists*			*Cumulative* Statistics	
Clock					*Loader*	*Weigh*	*Future Event*		
t	$LQ(t)$	$L(t)$	$WQ(t)$	$W(t)$	*Queue*	*Queue*	*List*	B_L	B_S
0	3	2	0	1	$DT4$		$(EL, 5, DT3)$	0	0
					$DT5$		$(EL, 10, DT2)$		
					$DT6$		$(EW, 12, DT1)$		
5	2	2	1	1	$DT5$	$DT3$	$(EL, 10, DT2)$	10	5
					$DT6$		$(EL, 5+5, DT4)$		
							$(EW, 12, DT1)$		
10	1	2	2	1	$DT6$	$DT3$	$(EL, 10, DT4)$	20	10
						$DT2$	$(EW, 12, DT1)$		
							$(EL, 10+10, DT5)$		
10	0	2	3	1		$DT3$	$(EW, 12, DT1)$	20	10
						$DT2$	$(EL, 20, DT5)$		
						$DT4$	$(EL, 10+15, DT6)$		
12	0	2	2	1		$DT2$	$(EL, 20, DT5)$	24	12
						$DT4$	$(EW, 12+12, DT3)$		
							$(EL, 25, DT6)$		
							$(ALQ, 12+60, DT1)$		
20	0	1	3	1		$DT2$	$(EW, 24, DT3)$	40	20
						$DT4$	$(EL, 25, DT6)$		
						$DT5$	$(ALQ, 72, DT1)$		
24	0	1	2	1		$DT4$	$(EL, 25, DT6)$	44	24
						$DT5$	$(EW, 24+12, DT2)$		
							$(ALQ, 72, DT1)$		
							$(ALQ, 24+100, DT3)$		
25	0	0	3	1		$DT4$	(EW, 36, $DT2$)	45	25
						$DT5$	$(ALQ, 72, DT1)$		
						$DT6$	$(ALQ, 124, DT3)$		
36	0	0	2	1		$DT5$	$(EW, 36+16, DT4)$	45	36
						$DT6$	$(ALQ, 72, DT1)$		
							$(ALQ, 36+40, DT2)$		
							$(ALQ, 124, DT3)$		
52	0	0	1	1		$DT6$	$(EW, 52+12, DT5)$	45	52
							$(ALQ, 72, DT1)$		
							$(ALQ, 76, DT2)$		
							$(ALQ, 52+40, DT4)$		
							$(ALQ, 124, DT3)$		

Table 3.6 Continued

Clock	*System State*				*Lists*			*Cumulative* Statistics	
					Loader	*Weigh*	*Future Event*		
t	$LQ(t)$	$L(t)$	$WQ(t)$	$W(t)$	*Queue*	*Queue*	*List*	B_L	B_S
64	0	0	0	1			$(ALQ, 72, DT1)$ $(ALQ, 76, DT2)$ $(EW, 64 + 16, DT6)$ $(ALQ, 92, DT4)$ $(ALQ, 124, DT3)$ $(ALQ, 64 + 80, DT5)$	45	64
72	0	1	0	1			$(ALQ, 76, DT2)$ $(EW, 80, DT6)$ $(EL, 72 + 10, DT1)$ $(ALQ, 92, DT4)$ $(ALQ, 124, DT3)$ $(ALQ, 144, DT5)$	45	72
76	0	2	0	1			$(EW, 80, DT6)$ $(EL, 82, DT1)$ $(EL, 76 + 10, DT2)$ $(ALQ, 92, DT4)$ $(ALQ, 124, DT3)$ $(ALQ, 144, DT5)$	49	76

short simulation in Table 3.6, the utilizations are estimated as follows:

$$\text{average loader utilization} = \frac{49/2}{76} = 0.32$$

$$\text{average scale utilization} = \frac{76}{76} = 1.00$$

These estimates cannot be regarded as accurate estimates of the long-run "steady-state" utilizations of the loader and scale; a considerably longer simulation would be needed to reduce the effect of the assumed conditions at time 0 (five of the six trucks at the loaders) and to realize accurate estimates. On the other hand, if the analyst were interested in the so-called transient behavior of the system over a short period of time (say 1 or 2 hours), given the specified initial conditions, then the results in Table 3.6 can be considered representative (or constituting one sample) of that transient behavior. Additional samples can be obtained by conducting additional simulations, each one having the same initial conditions but using a different stream of random digits to generate the activity times.

Table 3.6, the simulation table for the dump truck operation, could have been simplified somewhat by not explicitly modeling the dump trucks as entities.

That is, the event notices could be written as (EL, t), and so on, and the state variables used to keep track merely of the number of trucks in each part of the system, not which trucks were involved. With this representation, the same utilization statistics could be collected. On the other hand, if mean "system" response time, or proportion of trucks spending more than 30 minutes in the "system," were being estimated, where "system" refers to the loader queue and loaders and the weigh queue and scale, then dump truck entities (DTi), together with an attribute equal to arrival time at the loader queue, would be needed. Whenever a truck left the scale, that truck's response time could be computed as current simulation time (t) minus the arrival-time attribute. This new response time would be used to update the cumulative statistics: $S =$ total response time of all trucks which have been through the "system" and $F =$ number of truck response times which have been greater than 30 minutes. This example again illustrates the notion that to some extent the complexity of the model depends on the performance measures being estimated. ▲

EXAMPLE 3.6 (The Dump Truck Problem Revisited)

The events and activities were identified in Example 3.5. Using the activity-scanning approach, the conditions for beginning each activity are:

Activity	*Condition*
Loading time	Truck is at front of loader queue, and at least one loader is idle.
Weighing time	Truck is at front of weigh queue and weigh scale is idle.
Travel time	Truck has just completed weighing.

Using the process-interaction approach, we view the model from the viewpoint of one dump truck and its "life cycle." Considering a life cycle beginning at the loader queue, a dump truck process is pictured in Figure 3.8. ▲

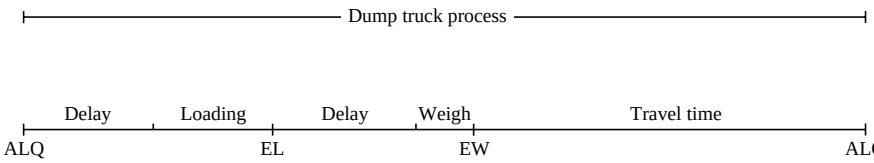


Figure 3.8 The dump truck process.

3.2 List Processing

List processing deals with methods for handling lists of entities and the future event list. Simulation packages provide, both explicitly for an analyst's use as well as hidden in the simulation mechanism behind the language, facilities for an analyst or the model itself to use lists and perform the basic operations on lists.

Section 3.2.1 describes the basic properties and operations performed on lists. Section 3.2.2 discusses the use of arrays for processing lists and the use of array indices to create linked lists, arrays being a simpler mechanism for describing the basic operations than the more general dynamically allocated linked lists discussed in Section 3.2.3. Finally, Section 3.2.4 briefly introduces some of the more advanced techniques for managing lists.

The purpose of this discussion of list processing is not to prepare the reader to implement lists and their processing in a general-purpose language such as FORTRAN, C or C++ but rather to increase the reader's understanding of lists and the underlying concepts and operations.

3.2.1 Lists: Basic properties and operations

As previously discussed, lists are a set of ordered or ranked records. In simulation, each record represents one entity or one event notice.

Since lists are ranked, they have a *top* or *head* (the first item on the list); some way to traverse the list (to find the second, third, etc. items on the list); and a *bottom* or *tail* (the last item on the list). A *head pointer* is a variable that points to or indicates the record at the top of the list. Some implementations of lists may also have a *tail pointer* that points to the bottom item on the list.

For purposes of discussion, an entity along with its attributes or an event notice will be referred to as a *record*. An entity identifier and its attributes are *fields* in the entity record; the event type, event time, and any other event-related data are fields in the event-notice record. Each record on a list will also have a field that holds a "next pointer" that points to the next record on the list, providing a way to traverse the list. Some lists may also require a "previous pointer" to allow traversing the list from bottom to top.

For either type of list, the main activities in list processing are adding a record to a list and removing a record from a list. More specifically, the main operations on a list are:

1. Removing a record from the top of the list.
2. Removing a record from any location on the list.
3. Adding an entity record to the top or bottom of the list.
4. Adding a record to an arbitrary position on the list, determined by the ranking rule.

While the first and third operations, removing or adding a record to the top or bottom of the list, can be carried out in minimal time by adjusting two record pointers and the head or tail pointer, the other two operations require at least a partial search through the list. Making these two operations efficient is the goal of list-processing techniques.

In the event-scheduling approach, when time is advanced and the imminent event is due to be executed, the removal operation takes place first,

namely, the event at the top of the FEL is removed from the FEL. If an arbitrary event is being canceled, or an entity is removed from a list based on some of its attributes (say, for example, its priority and due date) to begin an activity, then the second removal operation is performed. When an entity joins the back of a first-in first-out queue implemented as a list, then the third operation, adding an entity to the bottom of a list, is performed. Finally, if a queue has a ranking rule of earliest due date first, then upon arrival to the queue, an entity must be added to the list, not at the top or bottom, but at the position determined by the due-date ranking rule.

When simulating on a computer, whether using a general-purpose language such as FORTRAN, C or C++, or a simulation package, each entity record and event notice is stored in a physical location in computer memory. There are two basic possibilities: (a) All records are stored in arrays. Arrays hold successive records in contiguous locations in computer memory. They therefore can be referenced by an array index that can be thought of as a row number in a matrix. (b) All entities and event notices are represented by structures (as in C) or classes (as in C++), allocated from RAM as needed, and tracked by pointers to a record or structure.

Most simulation packages use dynamically allocated records and pointers to keep track of lists of items. As arrays are conceptually simpler, the concept of linked lists is first explained through arrays and array indices in Section 3.2.2 and then applied to dynamically allocated records and pointers in Section 3.2.3.

3.2.2 Using arrays for list processing

The array method of list storage is typical of FORTRAN but may be used in other procedural languages. As most versions of FORTRAN do not have actual record-type data structures, a record may be implemented as a row in a two-dimensional array or as a number of parallel arrays. For convenience, we use the notation R(i) to refer to the ith record in the array, however it may be stored in the language being used. Most modern simulation packages do not use arrays for list storage but rather use dynamically allocated records—that is, records that are created upon first being needed and destroyed when they are no longer needed.

Arrays are advantageous in that any specified record, say the ith, can be retrieved quickly without searching, merely by referencing R(i). Arrays are disadvantaged when items are added to the middle of a list or the list must be rearranged. In addition, arrays typically have a fixed size, determined at compile time or upon initial allocation when a program first begins to execute. In simulation, the maximum number of records for any list may be difficult or impossible to determine ahead of time, while the current number in a list may vary widely over the course of the simulation run. Worse yet, most simulations require more than one list, and if kept in separate arrays, each would have to be dimensioned to the largest the list would ever be, potentially using excessive amounts of computer memory.

When using arrays for storing lists, there are two basic methods for keeping track of the ranking of records in a list. One method is to store the first record in R(1), the second in R(2), and so on, and the last in R(*tailptr*), where *tailptr* is used to refer to the last item in the list. While simple in concept and easy to understand, this method will be extremely inefficient for all except the shortest lists of less than five or so records. For when adding an item, for example in position 41 in a list of 100 items, the last 60 records must be physically moved down one array position to make space for the new record. Even if it were a first-in first-out list, removing the top item would be inefficient, as all remaining items would have to be physically moved up one position in the array. The physical rearrangement method of managing lists will not be further discussed. What is needed is a method to track and rearrange the logical ordering of items in a list without having to physically move the records in computer memory.

In the second method, a variable called a head pointer, with name *headptr*, points to the record at the top of the list. For example, if the record in position R(11) was the record at the top of the list, then *headptr* would have a value of 11. In addition, each record has a field that stores the index or pointer of the next record in the list. For convenience, let R(i, next) represent the next index field.

EXAMPLE 3.7 (A List for the Dump Trucks at the Weigh Queue)

In Example 3.5, the dump truck problem, suppose that a waiting line of three dump trucks occurred at the weigh queue, specifically, $DT3$, $DT2$, and $DT4$, in that order, exactly as at CLOCK time 10 in Table 3.6. Suppose further that the model is tracking one attribute of each dump truck, its arrival time at the weigh queue, updated each time it arrives. Finally, suppose that the entities are stored in records in an array dimensioned from 1 to 6, one record for each dump truck. Each entity is represented by a record with 3 fields, the first an entity identifier, the second the arrival time at the weigh queue, and the last a pointer field to "point to" the next record, if any, in the list representing the weigh queue, as follows:

$$[\ \text{DT}i\text{, arrival time at weigh queue, next index}\ ]$$

Before their first arrival to the weigh queue, and before being added to the weigh queue list, the second and third fields are meaningless. At time 0, the records would be initialized as follows:

$$\begin{aligned}
\text{R}(1) &= [\ DT1, 0.0, 0] \\
\text{R}(2) &= [\ DT2, 0.0, 0] \\
&\vdots \\
\text{R}(6) &= [\ DT6, 0.0, 0]
\end{aligned}$$

Then, at CLOCK time 10 in the simulation in Table 3.6, the list of entities in the weigh queue would be defined by:

$$\text{headptr} = 3$$
$$\text{R}(1) = [\ DT1, 0.0, 0]$$
$$\text{R}(2) = [\ DT2, 10.0, 4]$$
$$\text{R}(3) = [\ DT3, 5.0, 2]$$
$$\text{R}(4) = [\ DT4, 10.0, 0]$$
$$\text{R}(5) = [\ DT5, 0.0, 0]$$
$$\text{R}(6) = [\ DT6, 0.0, 0]$$
$$\text{tailptr} = 4$$

To traverse the list, start with the head pointer, go to that record, retrieve that record's next pointer, and proceed, to create the list in its logical order, as for example:

$$\text{headptr} = 3$$
$$\text{R}(3) = [\ DT3, 5.0, 2]$$
$$\text{R}(2) = [\ DT2, 10.0, 4]$$
$$\text{R}(4) = [\ DT4, 10.0, 0]$$

The zero entry for next pointer in R(4), as well as tailptr $= 4$, indicates that $DT4$ is at the end of the list.

Using next pointers for a first-in first-out list, such as the weigh queue in this example, the operations of adding and removing entity records, as dump trucks join and leave the weigh queue, are particularly simple. At CLOCK time 12, dump truck $DT3$ begins weighing and thus leaves the weigh queue. To remove the $DT3$ entity record from the top of the list, update the head pointer by setting it equal to the next pointer value of the record at the top of the list, as in:

$$\text{headptr} = \text{R}(\text{headptr}, \text{next})$$

In this example, we get

$$\text{headptr} = \text{R}(3, \text{next}) = 2$$

meaning that dump truck $DT2$ in R(2) is now at the top of the list.

Similarly, at CLOCK time 20, dump truck $DT5$ arrives to the weigh queue and joins the rear of the queue. To add the $DT5$ entity record to the bottom of the list, the following steps are taken:

$$\text{R}(\text{tailptr}, \text{next}) = 5(\text{update the next pointer field of the previously last item})$$
$$\text{tailptr} = 5(\text{update the value of the tail pointer})$$

It becomes slightly more complex when a list is a ranked list, such as the future event list or an entity list ranked by an entity attribute. For ranked lists, to add or remove an item anywhere except to the head or tail of the list, searching is usually required. See Example 3.8.

Note that in the dump truck problem, the loader queue could also be implemented as a list using the same six records and the same array, because each dump truck entity will be on at most one of the two lists, and while loading, weighing or traveling, a dump truck will be on neither list. ▲

3.2.3 Using dynamic allocation and linked lists

In procedural languages such as C and C++, and in most simulation languages, entity records are dynamically created when an entity is created and event notice records are dynamically created whenever an event is scheduled on the future event list. The languages themselves, or the operating systems on which they are running, maintain a linked list of free chunks of computer memory and allocate a chunk of desired size upon request to running programs. (Another use of linked lists!) When an entity "dies"—that is, exits from the simulated system—and also after an event occurs and the event notice is no longer needed, the corresponding records are freed, making that chunk of computer memory available for later reuse; the language or operating system adds the chunk to the list of free memory.

In this text, we are not concerned with the details of allocating and freeing computer memory, and we will assume that the necessary operations occur as needed. With dynamic allocation, a record is referenced by a pointer instead of an array index. When a record is allocated in C or C++, the allocation routine returns a pointer to the allocated record, which must be stored in a variable or a field of another record for later use. A pointer to a record can be thought of as the physical or logical address in computer memory of the record.

In our example, we will use a notation for records identical to that in the previous section (3.2.2):

Entities:	[ID, attributes, next pointer]
Event notices:	[event type, event time, other data, next pointer]

but we will not reference them by the array notation R(i) as before, because it would be misleading. If for some reason we wanted the third item on the list, we would have to traverse the list, counting items until we reached the third record. Unlike arrays, there is no way to retrieve directly the ith record in a linked list, as the actual records may be stored at any arbitrary location in computer memory and are not stored contiguously as are arrays.

EXAMPLE 3.8 (The Future Event List and the Dump Truck Problem)

Based on Table 3.6, event notices in the dump truck problem of Example 3.5 are expanded to include a pointer to the next event notice on the future event

list and can be represented by:

$$[\text{ event type, event time, } DTi, nextptr]$$

as, for example,

$$[\ EL, 10, DT3, nextptr]$$

where EL is the end loading event to occur at future time 10 for dump truck $DT3$, and the field *nextptr* points to the next record on the FEL. Keep in mind that the records may be stored anywhere in computer memory, and in particular are not necessarily stored contiguously. Figure 3.9 represents the future event list at CLOCK time 10 taken from Table 3.6. The fourth field in each record is a pointer value to the next record in the future event list.

The C and C++ languages, and other general-purpose languages, use different notation for referencing data from pointer variables. For discussion purposes, if R is a pointer to a record, then

$$\text{R}\rightarrow\text{eventtype, R}\rightarrow\text{eventtime, R}\rightarrow\text{next}$$

are the event type, the event time, and the next record for the event notice that R points to. For example, if R is set equal to the head pointer for the FEL at CLOCK time 10, then

R→eventtype = EW

R→eventtime = 12

R→next is the pointer for the second event notice on the FEL,

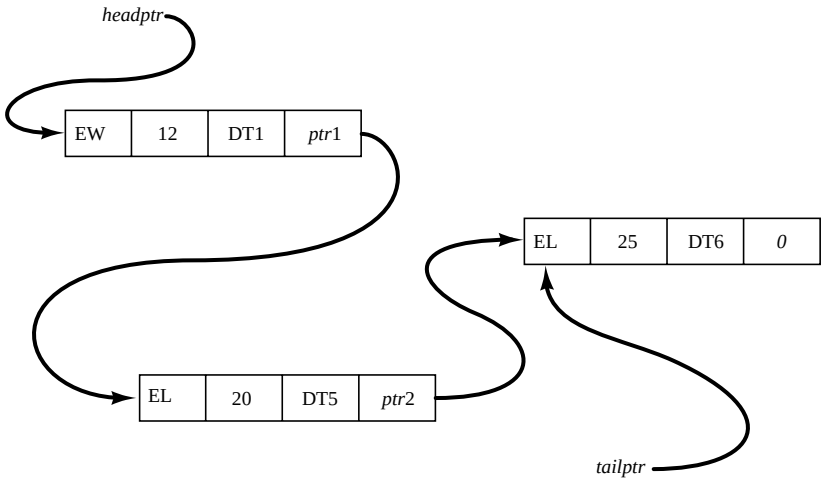


Figure 3.9 The dump truck future event list as a linked list.

so that

R→next→eventtype = EL

R→next→eventtime = 20

R→next→next is the pointer to the third event notice on the FEL.

If one of the pointer fields is zero (or null), then that record is the last item in the list, and the pointer variable *tailptr* points to that last record, as depicted in Figure 3.9. ▲

What we have described are called singly-linked lists, because there is a one-way linkage from the head of the list to its tail. The tail pointer is kept mostly for convenience and efficiency, especially for lists for which items are added at the bottom of the list. Of course, a tail pointer is not strictly necessary, as the last item can always be found by traversing the list, but it does make some operations more efficient.

For some purposes, it is desirable to traverse or search a list starting at the tail as well as from the head. For such purposes, a doubly-linked list can be used. Records on a doubly-linked list have two pointer fields, one for the next record and one for the previous record. Good references that discuss arrays, singly- and doubly-linked lists, and searching and traversing lists are Cormen et al. [1990] and Sedgewick [1998].

3.2.4 Advanced techniques

Many of the modern simulation packages use techniques and representations of lists that are more efficient than searching through a doubly-linked list. Most of these topics are too advanced for this text. The purpose of this section is to briefly introduce some of the more advanced ideas.

One idea to speed up processing doubly-linked lists is to use a middle pointer in addition to a head and tail pointer. With special techniques, the mid pointer will always point to the approximate middle of the list. Then, when a new record is being added to the list, the algorithm first examines the middle record to decide whether to begin searching at the head of the list or the middle of the list. Theoretically, except for some overhead due to maintenance of the mid pointer, this technique should cut search times in half. A few advanced techniques use one or more mid pointers, depending on the length of the list.

Some advanced algorithms use list representations other than a doubly-linked list, such as heaps or trees. These topics are beyond the scope of this text. Good references are Cormen et al. [1990] and Sedgewick [1998].

3.3 Summary

This chapter introduced the major concepts and building blocks in simulation, the most important being entities and attributes, events and activities. The three major world views—event scheduling, process interaction, and activity

scanning—were discussed. Finally, to gain an understanding of one of the most important underlying methodologies, Section 3.2 introduced some of the basic notions of list processing.

The next chapter will provide a survey of some of the most widely used and popular simulation packages, most of which either use exclusively or allow the process interaction approach to simulation.

REFERENCES

CORMEN, T. H., C. E. LEISEROON and R. L. RIVEST [1990], *Introduction to Algorithms*, McGraw-Hill, New York.

SCHRIBER, T. J. and D. T. BRUNNER [1998], "How Discrete-Event Simulation Software Works," in *Handbook of Simulation*, John Wiley, New York.

SEDGEWICK, R. [1998], *Algorithms in C++*, 3d ed., Addison-Wesley, Reading, MA.

EXERCISES

Instructions to the reader: For most exercises, the reader should first construct a model by explicitly defining:

1. System state
2. System entities and their attributes
3. Sets and the entities that may be put into the sets
4. Events and activities; event notices
5. Variables needed to collect cumulative statistics

Second, the reader should either (1) develop the event logic (as in Figures 3.5 and 3.6 for Example 3.3) in preparation for using the event-scheduling approach, or (2) develop the system processes (as in Figure 3.4) in preparation for using the process-interaction approach.

Most problems contain activities that are uniformly distributed over an interval $[a, b]$. When conducting a manual simulation, assume that $a, a+1, a+2, \ldots, b$ are the only possible values; that is, the activity time is a *discrete* random variable. The discrete assumption will simplify the manual simulation.

1 **(a)** Using the event-scheduling approach, continue the (manual) checkout-counter simulation in Example 3.3, Table 3.1. Use the same interarrival and service times that were previously generated and used in Example 2.1. When the last interarrival time is used, continue the simulation (allowing no new arrivals) until the system is empty. Compare the results obtained here to those obtained in Example 2.1. The results should be identical.

(b) Do Exercise 1(a) again, adding the model components necessary to estimate mean response time and proportion of customers who spend 4 or more minutes in the system. [*Hint:* See Example 3.4, Table 3.2.]

(c) Comment on the relative merits of manual versus computerized simulations.

2 Construct the event logic diagrams for the dump truck problem, Example 3.5.

3 In the dump truck problem of Example 3.5, it is desired to estimate mean response time and the proportion of response times which are greater than 30 minutes. A response time for a truck begins when that truck arrives at the loader queue, and ends when the truck finishes weighing. Add the model components and cumulative statistics needed to estimate these two measures of system performance. Simulate for 8 hours.

4 Prepare a table in the manner of Table 3.2, until the CLOCK reaches time 15, using the interarrival and service times given below in the order shown. The stopping event will be at time 30.

Interarrival times 1 5 6 3 8
Service times 3 5 4 1 5

5 Continue Table 3.2 until customer C5 departs.

6 The data for Table 3.2 are changed to the following:

Interarrival times 4 5 2 8 3 7
Service times 5 3 4 6 2 7
Prepare a table in the manner of Table 3.2 with a stopping event at time 25.

7 Redo Example 2.2 (the Able–Baker carhop problem) by a manual simulation using the event-scheduling approach.

8 Redo Example 2.4 (the (M, N) inventory system) by a manual simulation using the event-scheduling approach.

9 Redo Example 2.5 (the bearing-replacement problem) by a manual simulation using the event-scheduling approach.

10 Rework Example 3.5 using the following data:

Loading times 10 5 10 10 5 10 5
Weigh times 12 16 12 12 16 12 12
Travel times 40 60 40 80 100 40