José M. Garrido

Object Oriented Simulation

A Modeling and Programming Perspective

Springer

Chapter 1
Models and Simulation

1.1 Introduction

This chapter introduces the general concepts of modeling systems and simulation. The basic structure and behavior of systems and models are discussed. Other important topics discussed are: the various stages of a complete simulation study, the performance of a system estimated by running the simulation model, the general approaches for simulation modeling, and the simulation support software available.

1.2 Systems and Models

A system is composed of a collection of components or *entities* that interact and exhibit some type of behavior and together achieve a specific goal. The system usually interacts with its environment, which is the part of the real world that surrounds but is not part of the system. The system can be described as separated from the rest of its environment.

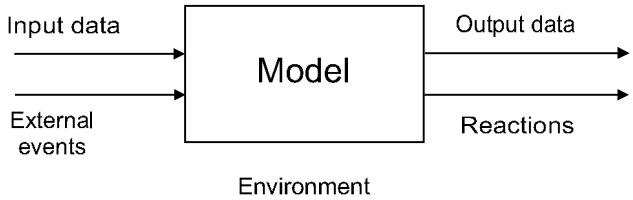

Fig. 1.1 A high-level black-box model of a system

A *model* is an abstract representation of a real system. The model is simpler than the real system, but it should be equivalent to the real system in all relevant as-

J.M. Garrido, *Object Oriented Simulation: A Modeling and Programming Perspective*,
DOI: 10.1007/978-1-4419-0516-1_1,

pects. The task of developing a model of a system is known as *modeling*. Figure 1.1 illustrates a general black-box model of a system surrounded by its environment.

A model is built to gain more and better understanding about the system it represents. Every model has a specific purpose and goal. A model includes only those aspects of the real system that were decided as being important, according to the initial requirements of the model. Therefore, the limitations of the model have to be clearly understood and documented.

The relevant structure and behavior of a real system are very conveniently represented in its model using object-oriented modeling techniques.

Abstraction is used in modeling and involves including in the model only the important aspects of the system. This results in less detail in the model than in the real system. Abstraction is very useful in modeling large and complex systems.

There are many types of models of interest; the most general categories of models are:

- Physical models
- Graphical models
- Mathematical models

Mathematical models are those in which the components and their relationships and behavior of the real system are represented by mathematical expressions. These models are very flexible, convenient, and powerful to use in problem solving. The types of models studied in this book – simulation models – belong to the category of mathematical models.

1.3 Simulation

Simulation is a set of techniques, methods, and tools for developing a simulation model of a system and using the simulation model to study the system.

The purpose of simulation is develop a simulation model, run experiments with the simulation model, and to gain better and understanding about the behavior of the real system that the model represents.

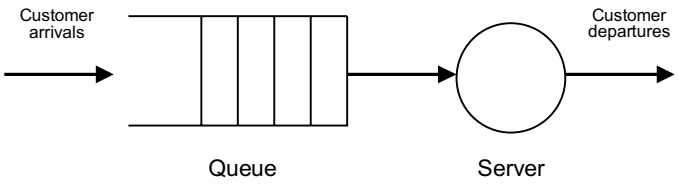

Fig. 1.2 A simple model of a single-server system

Figure 1.2 illustrates the high-level representation of a simple system that has arriving customers, a queue, and a server. An example of such a system is a barbershop with only one barber.

A dynamic system exhibits behavior that evolves over time; this includes various state changes at different points in time. The state of a system is the set of variables that are necessary to describe the system at a particular time. A (dynamic) system has:

- Structure
- Behavior

As previously mentioned, the system behavior depends on the inputs from the environment. For a simulation model to be useful, the model should allow the user to:

- Manipulate the model by supplying it with a corresponding set of inputs
- Observe its behavior or output
- Predict the behavior of the real system from the behavior of the model, under the same circumstances.

Most practical systems are very large and complex. The study of the dynamic behavior and the performance of such systems are very difficult. The development of a simulation model for studying a real-world system has two main purposes:

1. To study some relevant aspects by observing the operation of the model from the simulation runs
2. To estimate various performance measures

The output of the simulation model depends on its reaction to the following types of input:

- The passage of time
- Data from the environment
- Events (signals) from the environment.

The various components in a simulation model are usually classified in the following manner:

- *Flow entities*, these represent entities that flow through the system. These entities enter the system, are processed by various service components in the system, and exit at some other part of the system. The flow entities in the model represent the *customers* in the system as shown in Figure 1.2.
- *Servers*, these entities represent service components that provide some type of processing (or service) to the flow entities.
- *Queues*, these objects are used to place flow entities waiting for service or for one or more resources. Usually, there are one or more queues for each server.
- *Resources*, these are objects that can be acquired (or seized) and released by the flow entities.

Some of the terms discussed are not used universally; different terms are used in the various simulation support software and are not entirely standard. In FlexSim, the flow entities are known as *flowitems*. In Arena, a flow entity is known simply as an *entity*. A server is simply known as a resource in both Arena and Flexsim.

1.3.1 Computer Simulation Models

A simulation model is one implemented as a set of procedures that when executed in a computer, *mimic* the behavior (in some relevant aspects) and the static structure of the real system. This type of models uses empirical methods as possibly the only way to achieve a solution. Simulation models include as much detail as necessary; the representation of arbitrary complexity. These models can be much more accurate than analytical models because potentially, any desired level of detail can be achieved in the solution. Figure 1.3 shows a model of a multi-server system.

The general purpose of a simulation model is to study the dynamic behavior of a system, i.e., the state changes of the model as time advances. The state of the model is defined by the values of its attributes, which are represented by state variables. For example, the number of waiting customers to be processed by a simple barbershop is represented as a state variable (an attribute), which changes its value with time. Whenever this attribute changes value, the system changes its state.

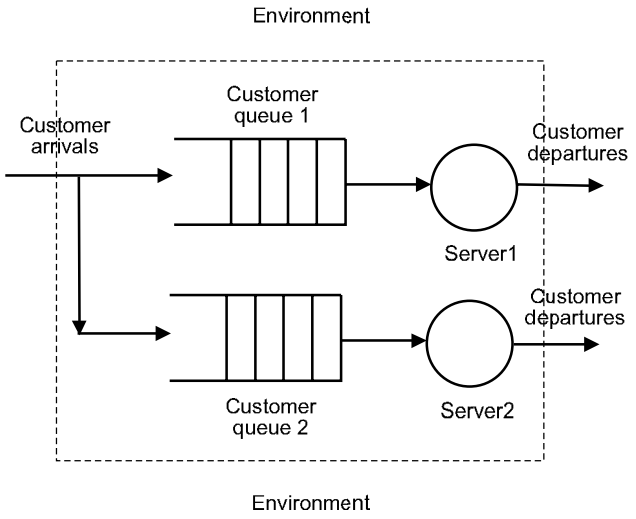

Fig. 1.3 A model of a multi-server system

There are several reasons for developing a simulation model and carrying out simulation runs:

- It may be too difficult, dangerous, and/or expensive to experiment with the real system.
- The real system is non-existing; simulations are used to study the behavior of a future system (to be built).

Simulation is applied in a wide variety of areas, some of these are:

- Computer systems
- Networks
- Systems security
- Medical systems
- Complex defense systems and war-gaming
- Stock market
- Weather prediction
- Computational fluid dynamics
- Traffic management
- Manufacturing
- Astrophysics
- Molecular dynamics
- Business enterprise
- Equipment training

1.3.2 Simulation Results

A simulation run is an experiment carried on the simulation model for some period of observation; the time dimension is one of the most important in simulation. Several simulation runs are usually necessary; in order to achieve some desired solution.

The results of experimenting with simulation models, i.e., simulation runs, can be broken down into two sets of outputs:

1. Trace of all the events that occur during the simulation period and all the information about the state of the model at the instants of the events; this directly reflects the dynamic behavior of the model
2. Performance measures (or metrics), which are the summary results of the simulation run.

The trace allows the users to verify that the model is actually interacting in the manner according to the model's requirements. The performance measures are the outputs that are analyzed for estimates used for capacity planning or for improving the current real system.

1.3.3 Models with Random Behavior

Models that are solved by empirical (or numeric) methods can be further divided into two categories:

1. Deterministic models
2. Stochastic models

A deterministic model displays a completely predictable behavior. A stochastic model includes some uncertainty implemented with random variables, whose values follow a probabilistic distribution. Most simulation models are stochastic because the real systems being modeled usually exhibits inherent uncertainty properties.

A model of a simple barbershop in which all the customers arrive at exact instants, and all have exact specified execution periods of service, is a *deterministic* simulation model because the behavior of the model can be completely and exactly determined.

A model with randomly varying arriving instances, varying service demand from each customer, is a *stochastic* model if only the averages of these parameters are specified together with a probability distribution for the variability of these parameters (workload) in the model. Uncertainty is included in this model because these parameter values cannot be exactly determined.

1.3.4 Discrete-Event Simulation Models

Simulation models can be divided into two general categories:

1. Continuous models
2. Discrete-event models

A continuous model is one in which the changes of state in the model occur continuously with time. Often the state variables in the model are represented as continuous functions of time.

For example, a model that represents the temperature in a boiler as part of a power plant can be considered a continuous model because the state variable that represents the temperature of the boiler is implemented as a continuous function of time. These types of models are usually deterministic and are modeled as a set of partial differential equations.

A discrete-event model is one representing a system that changes its states at discrete points in time, i.e., at specific instants. The simple model of a barbershop is a discrete-event model because when an arrival event occurs, there is a change in the state variable that represents the number of customers waiting to receive service from the barber (the server). This state variable and any other only change state when an event occurs, at discrete instants.

1.4 Input Parameters and Performance

The performance study of a system gives an indication of how well (or badly) is the system carrying out its functions, considering some particular aspect. A performance metric is a measure related to the efficiency and effectiveness of the system. Usually, a set of several different performance measures is necessary, one for every aspect considered.

There are three approaches that can be followed in studying the performance of systems:

1. Measurements
2. Simulation models
3. Analytical (mathematical) models

The last two approaches require the development of performance models. Performance modeling involves the construction of a simulation model or a mathematical analytical model for the purpose of determining some specified performance measures.

A complete performance study includes the definition of the following components:

- A set of relevant objectives and goals
- The performance metrics
- The system parameters
- The system factors
- The system workloads

The capacity planning of any system is determined by its maximum performance. Functionality is the ability of the system to perform correctly. In today's complex environment, the use and demand of computers and computer networks is increasing at an extremely rapid rate, so the capacity planning and performance modeling are very important for the design and operation of these systems.

1.4.1 Workload and System Parameters

The workload consists of a list of service requests to the system. Predicting the values of the performance metrics depends not only on the workload parameters but also on the system parameters.

Examples of workload parameters are:

- Average service demand of customers
- Average arrival rate of customers
- Average resource demands of the customers

Examples of system parameters are: customer queue size, number of server queues, server speed, total number and types of resources available in the system.

1.4.2 Performance Metrics

The main results of running a performance model are the values of the performance measures. The most common performance measures are:

- Average response times - the average period between an arrival of a customer and the instant that the customer has received service.
- Average throughput - the average number of customers serviced in some specified period.
- server utilization - the portion of total (simulation) time that the server carries out work (usual a percentage); usually, this is percentage of time that the server is not idle.

Other performance measures usually relevant in studying operating systems are:

- Average waiting time
- Availability
- Reliability
- Capacity
- Fairness
- Speedup

1.4.3 Additional Performance Concepts

Parameters that changed during a performance evaluation are called factors, for example, the number of users.

The general goal of most performance studies is one of the following:

- Compare different alternatives
- Find the optimal parameter value

The nominal capacity of a system is given by the maximum achievable throughput under ideal workload conditions. The usable capacity is the maximum throughput achievable under given constraints. The efficiency of a system is the ratio of usable capacity to nominal capacity.

The reliability of a system is measured by the probability of errors (or by the mean time between errors). The availability is given by the fraction of the time that the system is available for user requests. The interval during which the system is available is called the uptime. Often the mean uptime is called the mean time to failure (MTTF).

1.5 Modeling System Behavior

Dynamic and stochastic systems are systems whose behavior changes with time and that include some degree of uncertainty. The state of a *dynamic system* is usually expressed as a function of time (time dependent). An operation is a sequence of activities that may change the state of the system. An *event* is the occurrence at an instance of time.

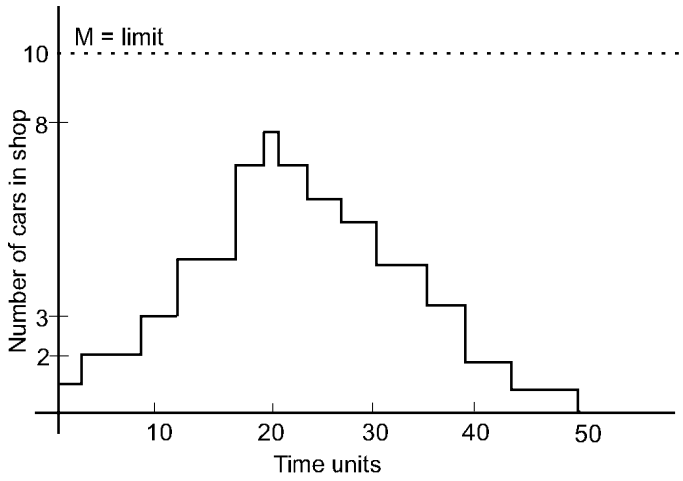

Fig. 1.4 Number of cars in the queue

The model of a simple carwash system is a discrete-event model; when an arrival event occurs, it causes a change in the state variable that represents the number of cars waiting to receive service from the machine (the server). This state variable and any other only change its value when an event occurs, i.e., at discrete instants. Figure 1.4 shows the changes in the number of cars in the model for the simple car-wash system.

The dynamic composition of a system can be described in terms of activities, events, and processes (active objects). An activity is the smallest unit of work; it has a finite interval of time to perform. A sequence of activities (an operation) is initiated when an event occurs. One or more activities transform the state of a system.

Processes are active components of a model and are used to represent the structure and all or part of the life of active entities in the real system.

1.6 Simulation Time

In addition to the components in a model, the simulation software has a *simulation clock*. This maintains the simulation time and is updated after every event in a *simulation run.*

The simulation time at which an event occurs is called its *event time*. The simulation executive, (i.e., the program that implements and controls the simulation) must include a means for carrying out a time change in the system and must keep track of the passage of simulation time. Increasing the value of the *simulation clock* achieves this. There are two basic techniques to change the time of the simulation clock:

- Fixed time increment, or periodic scan
- Variable time increment or event scan

Simulation executives that use fixed time increments are also called synchronous executives. The value of the simulation clock is incremented by a fixed amount, which is a predetermined uniform unit. After the simulation clock is adjusted by this fixed time increment, the system is examined to determine whether any events occurred during that interval. If any events occurred, they are simulated. The simulation clock is then advanced another time unit, and the cycle is repeated. These simulation executives (also called simulators) are, in general, less efficient than simulators that use variable time increments.

The main limitation of this handling of time is that an event may occur between two time increments, in which case the simulator simulates this event as if it had occurred at the end of the interval. In addition to this, the system needs to examine the simulation at fixed intervals, in some cases to find out that nothing has happened.

Variable time increment of the simulation clock is used by most discrete-event systems and offers the greatest flexibility. The simulation clock is incremented at the occurrence of the next imminent event. Thus, the simulation clock is incremented from one event time to the next event time, without regarding the interval that separates these event occurrences. After updating the simulation clock, the simulation system simulates the event (implements the resulting changes), and the whole cycle is repeated.

In a typical system, there may be several activities being carried out in parallel (simultaneously). The two approaches to handling time, periodic scan and event scan, are simple approaches to simulating parallelism.

1.7 Discrete-Event Simulation Approaches

There are three world views (approaches) for discrete-event simulation. The various simulation systems provide facilities for model implementation according to the world view. The three approaches are:

1. Activity scanning

2. Event scheduling
3. Process interaction

During a simulation run, the simulation executive (simulation control program) and the software model execute together. The software model (the program that implements the simulation model) is composed of a number of code segments whose structure depends on the simulation approach used.

1.7.1 The Activity World View

The activity is the basic building block of the activity world view. An activity sequence immediately follows a state change in the system. In the model program, each activity sequence is considered a segment of program waiting to be executed.

An activity sequence is executed if and only if the corresponding conditions are satisfied. At the end of each activity sequence, control is returned to the simulation executive. If the conditions corresponding to an activity sequence are not satisfied, then control is returned to the simulation executive without performing the activity sequence. The system moves from event to event, and at each event, the conditions for each activity sequence are tested. The activity scanning approach provides locality of state; each activity routine in a model describes all the activities that must be performed due to the model assuming a particular state (i.e., because of a particular set of conditions becoming true).

1.7.2 The Event World View

The event is the building block for the event world view. The code segments of the software model are event routines. There is an event routine associated with each type of event, and it performs the operations required to handle that type of event. An event routine is invoked (or called) every time an event of that type occurs. An event routine is composed of the set of operations or activities that may follow from a state change in the model. The simulation executive uses the event list to find the next imminent event and then invokes the corresponding event routine.

1.7.3 The Process Interaction World View

The major components of a system are known as *entities*. There are two important categories of these entities, the first category are the *active* entities (i.e., they have a life of their own). These active entities are also known as *process*. The other category includes the *passive* entities.

A system is composed of active and passive components or entities. Active entities interact among themselves and with the *environment* to accomplish the system's goal. This interaction determines the behavior of the system. These active entities are known as processes.

In the process world view, the behavior of the system under study is represented by a set of interacting processes. A process performs a sequence of operations during its life within the system. The merging of the event sequences of these processes contains all events that occur in the system.

The *event list* used with this approach is composed of a sequence of event nodes (also called event notices), each of which contains the event time and the process (or pointer to the process) to which the event belongs. This sequence is ordered by non-decreasing values of the event times. Similar to the other approaches to discrete-event simulation, the simulation executive manages the processes in the event list. The simulation executive carries out the following tasks:

1. Placement of processes at particular points in time in the event list
2. Removal of processes from the event list
3. Rescheduling of processes (in the event list)

The simulation executive keeps track of simulation time with the clock, and from the event list, decides which process to activate next. The event time for this process (the current process) becomes the new value of the simulation clock. The operations of the current process are performed after the simulation executive advances the clock.

1.8 Simulation Software

Although all that is required for implementing a simulation model can be accomplished with a general-purpose high-level programming language, it is much easier and convenient to use some type of simulation software. The various levels of simulation support software are:

1. A general-purpose (object-oriented) programming language, such as C++ and Java
2. A simulation package or library, such as PsimJ2, Silk, and Simjava, Psim3. These are used with a general-purpose programming language
3. An object-oriented simulation language, such as OOSimL and Simscript III
4. An integrated simulation environment, such as Arena, FlexSim, Simul8, and ProModel.

Simulation software at levels 2 and 3 provide the most flexibility and power for implementing simulation models. The last level (level 4) provides the easiest and simplest way for developing simulation models but at the expense of flexibility and ability to implement fairly large and complex models. Most simulation software tools provide the following functions:

- Advancing the simulation clock when an event occurs
- Scheduling the events by their time of occurrence
- Providing simple and priority queues
- Providing standard mechanism for resource manipulation by processes
- Providing the general behavior for processes (active objects)
- Providing random number generators for various probability distributions

The following list includes several relevant software tools for discrete-event simulation.

SimPack and Sim++	GPSS	CSIM
FlexSim	SLAM	Simple++
Taylor II	SimPy	Arena
Extend	AweSim	iThink
Simul8	DESP	ProModel

The following list includes several software packages for discrete-event simulation for use with the Java programming language.

JavaSim	JSIM	DESMO-J
Simjava	Silk	PsimJ2

The following list includes several simulation languages for discrete-event simulation.

Simula/DEMOS	GPSS	Simscript 2.5
Modsim III	Simscript III	OOSimL

1.9 Simulation Study

A simulation study involves much more than developing a simulation model. This study consists of a sequence of stages that starts with the definition of the goals and is carried out in a possibly iterative manner. The simulation study is defined by the following general steps:

1. Definition of the *problem statement* for the simulation study. This statement must provide the description of the purpose for building the model, the questions it must help to answer, and the performance measures relevant to these questions.
2. *Data collection*. This stage includes tasks of gathering as much data as possible about the existing system under study, a similar system, and/or the environment where the new system will operate. The model parameters and input probabilities to be used in the model will be defined. Assumptions have to be taken on those aspects of the model for which no data is available.
3. Definition of the *conceptual model*. This is a description of what is to be accomplished with the simulation model to be constructed; ideally, the description should be clear, precise, complete, concise, and understandable. This description includes the list of relevant components, the interactions among the components, and the relationships among the components.
4. *Design of the model*. This stage describes the details of the data structures necessary to implement the components and the details of the algorithms for the relationships and dynamic behavior of the model.
5. *Implementation of the simulation Model*. This implementation can be carried out using a general-purpose high-level programming language, a simulation language, or an integrated software simulation environment. The main tasks in this phase are the coding, debugging, and testing the software model.
6. *Verification of the model*. From different runs of the implementation of the model (or the model program), this stage compares the output results with those that would have been produced by a correct implementation of the conceptual model.
7. *Validation of the model*. This stage compares the outputs of the verified model with the outputs of a real system. This stage compares the model with reality.
8. *Design of experiments*. This stage includes planning the experiments according to the goals of the simulation study.
9. *Analysis of results*. This stage involves statistical analysis and interpretation of the results and depends heavily on the previous stage.

The conceptual model of the system is usually described using the Unified Modeling Language (UML) and additional notation used in simulation modeling. UML is a standard set of graphical notational conventions used to describe various views about the structure and behavior of the system under study.

The conceptual model is formulated from the initial problem statement, informal user requirements, and data and knowledge gathered from analysis of previously developed models. Figure 1.5 illustrates the relationship between the conceptual model and the simulation model.

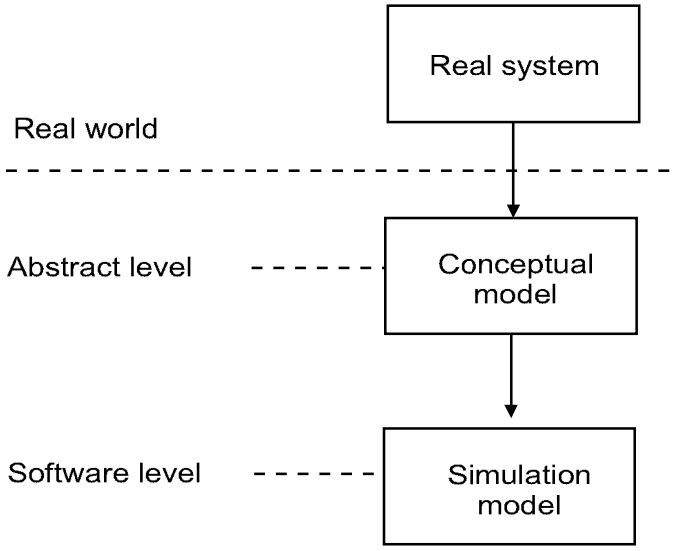

Fig. 1.5 Conceptual Model

1.10 Key Terms

simulation model	system	components
entities	environment	flow entities
servers	resources	queues
simulation run	trace	performance metric
random behavior	simulation study	verification
validation	simulation software	workload

1.11 Summary

A model is an abstract representation of a system. Simulation involves developing a simulation model of a system, carrying out experiments with the model, making inferences from the results of running the simulation model. A performance model is a model developed to study the performance of a system. The common types of simulation models are: continuous, discrete-event, deterministic, and stochastic. The performance evaluation of systems is often carried out with simulation models. Examples of performance measures are throughput, average response time, and server utilization.