The Discrete Event Simulation Approach

Introduction

Simulating a probabilistic model involves generating the stochastic mechanisms of the model and then observing the resultant flow of the model over time. Depending on the reasons for the simulation, there will be certain quantities of interest that we will want to determine. However, because the model's evolution over time often involves a complex logical structure of its elements, it is not always apparent how to keep track of this evolution so as to determine these quantities of interest. A general framework, built around the idea of "discrete events," has been developed to help one follow a model over time and determine the relevant quantities of interest. The approach to simulation based on this framework is often referred to as the *discrete event simulation approach*.

7.1 Simulation via Discrete Events

The key elements in a discrete event simulation are variables and events. To do the simulation we continually keep track of certain variables. In general, there are three types of variables that are often utilized—the time variable, and the system state variable.

Variables

1. Time variable t	This refers to the amount of (simulated) time that has elapsed
2. Counter variables	These variables keep a count of the number of times that certain events have occurred by time t
3. System state (**SS**) variable	This describes the "state of the system" at the time t

Simulation. DOI: http://dx.doi.org/10.1016/B978-0-12-415825-2.00007-3

Whenever an "event" occurs the values of the above variables are changed, or updated, and we collect, as output, any relevant data of interest. In order to determine when the next event will occur, an "event list," which lists the nearest future events and when they are scheduled to occur, is maintained. Whenever an event "occurs" we then reset the time and all state and counter variables and collect the relevant data. In this way we are able to "follow" the system as it evolves over time.

As the preceding is only meant to give a very rough idea of the elements of a discrete event simulation, it is useful to look at some examples. In Section 7.2 we consider the simulation of a single-server waiting line, or queueing, system. In Sections 7.3 and 7.4 we consider multiple-server queueing systems. The model of Section 7.3 supposes that the servers are arranged in a series fashion, and the one of 7.4 that they are arranged in a parallel fashion. In Section 7.5 we consider an inventory stocking model, in 7.6 an insurance risk model, and in 7.7 a multimachine repair problem. In Section 7.8 we consider a model concerning stock options.

In all the queueing models, we suppose that the customers arrive in accordance with a nonhomogeneous Poisson process with a bounded intensity function $\lambda(t), t > 0$. In simulating these models we will make use of the following subroutine to generate the value of a random variable T_s, defined to equal the time of the first arrival after time s.

Let λ be such that $\lambda(t) \leqslant \lambda$ for all t. Assuming that $\lambda(t), t > 0$, and λ are specified, the following subroutine generates the value of T_s.

A Subroutine for Generating T_s

STEP 1: Let $t = s$.
STEP 2: Generate U.
STEP 3: Let $t = t - \frac{1}{\lambda} \log U$.
STEP 4: Generate U.
STEP 5: If $U \leqslant \lambda(t)/\lambda$, set $T_s = t$ and stop.
STEP 6: Go to Step 2.

7.2 A Single-Server Queueing System

Consider a service station in which customers arrive in accordance with a nonhomogeneous Poisson process with intensity function $\lambda(t), t \geqslant 0$. There is a single server, and upon arrival a customer either enters service if this server is free at that moment or else joins the waiting queue if the server is busy. When the server completes serving a customer, it then either begins serving the customer that had been waiting the longest (the so-called "first come first served" discipline) if there are any waiting customers, or, if there are no waiting customers, it remains free until the next customer's arrival. The amount of time it takes to service a

customer is a random variable (independent of all other service times and of the arrival process), having probability distribution G. In addition, there is a fixed time T after which no additional arrivals are allowed to enter the system, although the server completes servicing all those that are already in the system at time T.

Suppose that we are interested in simulating the above system to determine such quantities as (a) the average time a customer spends in the system and (b) the average time past T that the last customer departs—that is, the average time at which the server can go home.

To do a simulation of the preceding system we use the following variables:

Time Variable	t
Counter Variables	N_A: the number of arrivals (by time t)
	N_D: the number of departures (by time t)
System State Variable	n: the number of customers in the system (at time t)

Since the natural time to change the above quantities is when there is either an arrival or a departure, we take these as the "events." That is, there are two types of event: arrivals and departures. The event list contains the time of the next arrival and the time of the departure of the customer presently in service. That is, the event list is

$$\mathbf{EL} = t_A, t_D$$

where t_A is the time of the next arrival (after t) and t_D is the service completion time of the customer presently being served. If there is no customer presently being served, then t_D is set equal to ∞.

The output variables that will be collected are $A(i)$, the arrival time of customer i; $D(i)$, the departure time of customer i; and T_p, the time past T that the last customer departs.

To begin the simulation, we initialize the variables and the event times as follows:

Initialize

Set $t = N_A = N_D = 0$.
Set $\mathbf{SS} = 0$.
Generate T_0, and set $t_A = T_0, t_D = \infty$.

To update the system, we move along the time axis until we encounter the next event. To see how this is accomplished, we must consider different cases, depending on which member of the event list is smaller. In the following, Y refers to a service time random variable having distribution G.

$$t = \text{timevariable}, \qquad \mathbf{SS} = n, \qquad \mathbf{EL} = t_A, t_D$$

Case 1: $t_A \leqslant t_D, t_A \leqslant T$

Reset: $t = t_A$ (we move along to time t_A).
Reset: $N_A = N_A + 1$ (since there is an additional arrival at time t_A).
Reset: $n = n + 1$ (because there is now one more customer).
Generate T_t, and reset $t_A = T_t$ (this is the time of the next arrival).
If $n = 1$, generate Y and reset $t_D = t + Y$ (because the system had been empty and so we need to generate the service time of the new customer).
Collect output data $A(N_A) = t$ (because customer N_A arrived at time t).

Case 1: $t_D < t_A, t_D \leqslant T$

Reset: $t = t_D$.
Reset: $n = n - 1$.
Reset: $N_D = N_D + 1$ (since a departure occurred at time t).
If $n = 0$, reset $t_D = \infty$; otherwise, generate Y and reset $t_D = t + Y$.
Collect the output data $D(N_D) = t$ (since customer N_D just departed).

Case 3: $\min(t_A, t_D) > T, n > 0$

Reset: $t = t_D$
Reset: $n = n - 1$
Reset: $N_D = N_D + 1$
If $n > 0$, generate Y and reset $t_D = t + Y$.
Collect the output data $D(N_D) = t$.

Case 4: $\min(t_A, t_D) > T, n = 0$

Collect output data $T_p = \max(t - T, 0)$.

The preceding is illustrated in the flow diagram presented in Figure 7.1. Each time we arrive at the "stop" box we would have collected the data N_A, the total number of arrivals, which will equal N_D, the total number of departures. For each $i, i = 1, \ldots, N_A$, we have $A(i)$ and $D(i)$, the respective arrival and departure times of customer i [and thus $D(i) - A(i)$ represents the amount of time that customer i spent in the system]. Finally, we will have T_p, the time past T at which the last customer departed. Each time we collect the above data we say that a simulation run has been completed. After each run we then reinitialize and generate another run until it has been decided that enough data have been collected. (In Chapter 8 we consider the question of when to end the simulation.) The average of all the values of T_p that have been generated will be our estimate of the mean time past T that the last customer departs; similarly, the average of all the observed values of $D - A$ (i.e., the average time, over all customers observed in all our simulation

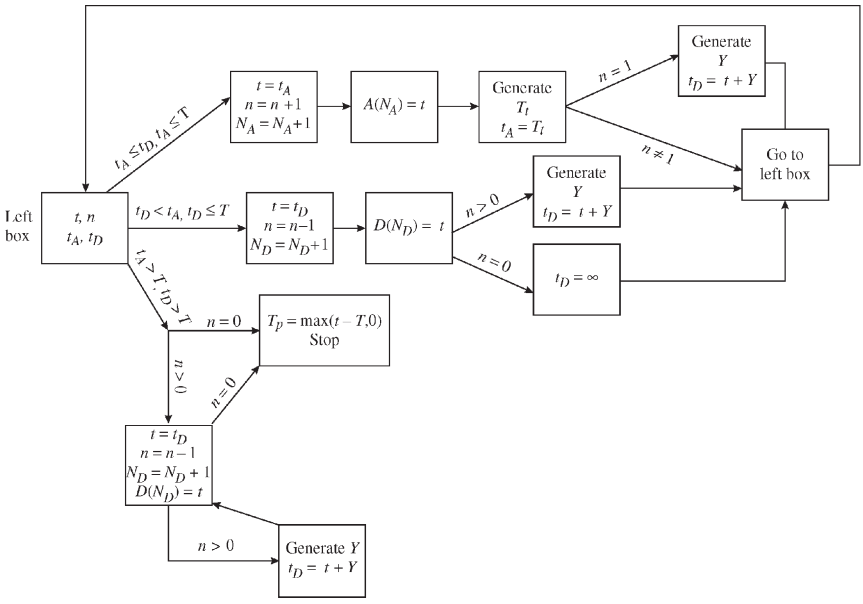

Figure 7.1. Simulating the Single Server Queue.

runs, that a customer spends in the system) will be our estimate of the average time that a customer spends in the system.

Remark If we want to save output data giving the number of customers in the system at each point of time, all that is necessary is to output the system state and time variable pair (n, t) whenever an event occurs. For instance, if the data (1, 4) and (0, 6) were output then, with $n(t)$ being the number in the system at time t, we would know that

$$\begin{aligned} n(t) &= 0, \quad \text{if } 0 \leqslant t < 4 \\ n(t) &= 1, \quad \text{if } 4 \leqslant t < 6 \\ n(t) &= 0, \quad \text{if } t = 6 \end{aligned}$$ □

7.3 A Queueing System with Two Servers in Series

Consider a two-server system in which customers arrive in accordance with a nonhomogeneous Poisson process, and suppose that each arrival must first be served by server 1 and upon completion of service at 1 the customer goes over to server 2. Such a system is called a *tandem* or *sequential* queueing system. Upon

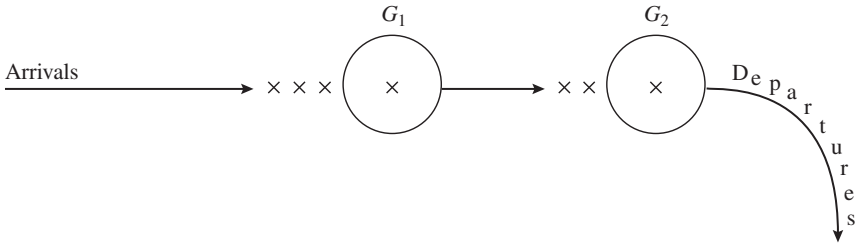

Figure 7.2. A Tandem Queue.

arrival the customer will either enter service with server 1 if that server is free, or join the queue of server 1 otherwise. Similarly, when the customer completes service at server 1 it then either enters service with server 2 if that server is free, or else it joins its queue. After being served at server 2 the customer departs the system. The service times at server i have distribution $G_i, i = 1, 2$. (See Figure 7.2.)

Suppose that we are interested in using simulation to study the distribution of the amounts of time that a customer spends both at server 1 and at server 2. To do so, we will use the following variables.

Time Variable t
System State (SS) Variable
(n_1, n_2): if there are n_1 customers at server 1 (including both those in queue and in service) and n_2 at server 2
Counter Variables
N_A: the number of arrivals by time t
N_D: the number of departures by time t
Output Variables
$A_1(n)$: the arrival time of customer $n, n \geqslant 1$
$A_2(n)$: the arrival time of customer n at server 2, $n \geqslant 1$
$D(n)$: the departure time of customer $n, n \geqslant 1$
Event List t_A, t_1, t_2, where t_A is the time of the next arrival, and t_i is the service completion time of the customer presently being served by server $i, i = 1, 2$. If there is no customer presently with server i, then $t_i = \infty, i = 1, 2$. The event list always consists of the three variables t_A, t_1, t_2.

To begin the simulation, we initialize the variables and the event list as follows:

Initialize

Set $t = N_A = N_D = 0$.
Set **SS** $= (0, 0)$.
Generate T_0, and set $t_A = T_0, t_1 = t_2 = \infty$.

To update the system, we move along in time until we encounter the next event. We must consider different cases, depending on which member of the event list is smallest. In the following, Y_i refers to a random variable having distribution $G_i, i = 1, 2$.

$$\mathbf{SS} = (n_1, n_2) \qquad \mathbf{EL} = t_A, t_1, t_2$$

Case 1: $t_A = \min(t_A, t_1, t_2)$

Reset: $t = t_A$.
Reset: $N_A = N_A + 1$.
Reset: $n_1 = n_1 + 1$.
Generate T_t, and reset $t_A = T_t$.
If $n_1 = 1$, generate Y_1 and reset $t_1 = t + Y_1$.
Collect output data $A_1(N_A) = t$.

Case 2: $t_1 < t_A, t_1 \leqslant t_2$

Reset: $t = t_1$.
Reset: $n_1 = n_1 - 1, n_2 = n_2 + 1$.
If $n_1 = 0$, reset $t_1 = \infty$; otherwise, generate Y_1 and reset $t_1 = t + Y_1$.
If $n_2 = 1$, generate Y_2 and reset $t_2 = t + Y_2$.
Collect the output data $A_2(N_A - n_1) = t$.

Case 3: $t_2 < t_A, t_2 < t_1$

Reset: $t = t_2$.
Reset: $N_D = N_D + 1$.
Reset: $n_2 = n_2 - 1$.
If $n_2 = 0$, reset $t_2 = \infty$.
If $n_2 > 0$, generate Y_2, and reset $t_2 = t + Y_2$.
Collect the output data $D(N_D) = t$.

Using the preceding updating scheme it is now an easy matter to simulate the system and collect the relevant data.

7.4 A Queueing System with Two Parallel Servers

Consider a model in which customers arrive at a system having two servers. Upon arrival the customer will join the queue if both servers are busy, enter service with server 1 if that server is free, or enter service with server 2 otherwise. When the customer completes service with a server (no matter which one), that customer then departs the system and the customer that has been in queue the longest (if there are any customers in queue) enters service. The service distribution at server i is $G_i, i = 1, 2$. (See Figure 7.3.)

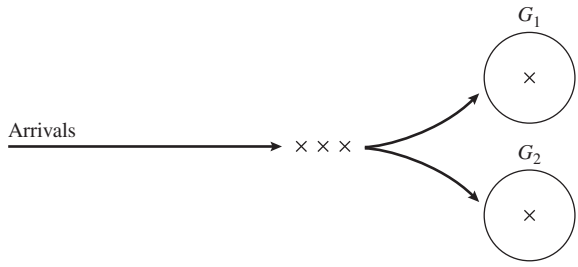

Figure 7.3. A Queue with Two Parallel Servers.

Suppose that we want to simulate the preceding model, keeping track of the amounts of time spent in the system by each customer, and the number of services performed by each server. Because there are multiple servers, it follows that customers will not necessarily depart in the order in which they arrive. Hence, to know which customer is departing the system upon a service completion we will have to keep track of which customers are in the system. So let us number the customers as they arrive, with the first arrival being customer number 1, the next being number 2, and so on. Because customers enter service in order of their arrival, it follows that knowing which customers are being served and how many are waiting in queue enables us to identify the waiting customers. Suppose that customers i and j are being served, where $i < j$, and that there are $n - 2 > 0$ others waiting in queue. Because all customers with numbers less than j would have entered service before j, whereas no customer whose number is higher than j could yet have completed service (because to do so they would have had to enter service before either i or j), it follows that customers $j + 1, \ldots, j + n - 2$ are waiting in queue.

To analyze the system we will use the following variables:

Time Variable t

System State Variable (SS)

(n, i_1, i_2) if there are n customers in the system, i_1 is with server 1 and i_2 is with server 2. Note that $\mathbf{SS} = (0)$ when the system is empty, and $\mathbf{SS} = (1, j, 0)$ or $(1, 0, j)$ when the only customer is j and he is being served by server 1 or server 2, respectively.

Counter Variables

N_A: the number of arrivals by time t

C_j: the number of customers served by j, $j = 1, 2$, by time t

Output Variables

$A(n)$: the arrival time of customer n, $n \geqslant 1$

$D(n)$: the departure time of customer n, $n \geqslant 1$

Event list t_A, t_1, t_2
where t_A is the time of the next arrival, and t_i is the service completion time of the customer presently being served by server $i, i = 1, 2$. If there is no customer presently with server i, then we set $t_i = \infty, i = 1, 2$. In the following, the event list will always consist of the three variables t_A, t_1, t_2.

To begin the simulation, we initialize the variables and event list as follows:

Initialize

Set $t = N_A = C_1 = C_2 = 0$.
Set $\mathbf{SS} = (0)$.
Generate T_0, and set $t_A = T_0, t_1 = t_2 = \infty$.

To update the system, we move along in time until we encounter the next event. In the following cases, Y_i always refers to a random variable having distribution $G_i, i = 1, 2$.

Case 1: $\mathbf{SS} = (n, i_1, i_2)$ and $t_A = \min(t_A, t_1, t_2)$

Reset: $t = t_A$.
Reset: $N_A = N_A + 1$.
Generate T_t and reset $t_A = T_t$.
Collect the output data $A(N_A) = t$.

If $\mathbf{SS} = (0)$:

Reset: $\mathbf{SS} = (1, N_A, 0)$.
Generate Y_1 and reset $t_1 = t + Y_1$.

If $\mathbf{SS} = (1, j, 0)$:

Reset: $\mathbf{SS} = (2, j, N_A)$.
Generate Y_2 and reset $t_2 = t + Y_2$.

If $\mathbf{SS} = (1, 0, j)$:

Reset $\mathbf{SS} = (2, N_A, j)$.
Generate Y_1 and reset $t_1 = t + Y_1$.

If $n > 1$:

Reset: $\mathbf{SS} = (n + 1, i_1, i_2)$.

Case 2: $\mathbf{SS} = (n, i_1, i_2)$ and $t_1 < t_A, t_1 \leqslant t_2$

Reset: $t = t_1$.
Reset: $C_1 = C_1 + 1$.

Collect the output data $D(i_1) = t$.

If $n = 1$:

Reset: $\mathbf{SS} = (0)$.
Reset: $t_1 = \infty$.

If $n = 2$:

Reset: $\mathbf{SS} = (1, 0, i_2)$.
Reset: $t_1 = \infty$.

If $n > 2$: Let $m = \max(i_1, i_2)$ and

Reset $\mathbf{SS} = (n - 1, m + 1, i_2)$
Generate Y_1 and reset $t_1 = t + Y_1$

Case 3: $\mathbf{SS} = (n, i_1, i_2)$ and $t_2 < t_A, t_2 < t_1$

The updatings in Case 3 are left as an exercise.

If we simulate the system according to the preceding, stopping the simulation at some predetermined termination point, then by using the output variables as well as the final values of the counting variables C_1 and C_2, we obtain data on the arrival and departure times of the various customers as well as on the number of services performed by each server.

7.5 An Inventory Model

Consider a shop that stocks a particular type of product that it sells for a price of r per unit. Customers demanding this product appear in accordance with a Poisson process with rate λ, and the amount demanded by each one is a random variable having distribution G. In order to meet demands, the shopkeeper must keep an amount of the product on hand, and whenever the on-hand inventory becomes low, additional units are ordered from the distributor. The shopkeeper uses a so-called (s, S) ordering policy; namely, whenever the on-hand inventory is less than s and there is no presently outstanding order, then an amount is ordered to bring it up to S, where $s < S$. That is, if the present inventory level is x and no order is outstanding, then if $x < s$ the amount $S - x$ is ordered. The cost of ordering y units of the product is a specified function $c(y)$, and it takes L units of time until the order is delivered, with the payment being made upon delivery. In addition, the shop pays an inventory holding cost of h per unit item per unit time. Suppose further that whenever a customer demands more of the product than is presently available, then the amount on hand is sold and the remainder of the order is lost to the shop.

Let us see how we can use simulation to estimate the shop's expected profit up to some fixed time T. To do so, we start by defining the variables and events as follows.

Time Variable t
System State Variable (x, y)
where x is the amount of inventory on hand, and y is the amount on order.
Counter Variables
C, the total amount of ordering costs by t
H, the total amount of inventory holding costs by t
R, the total amount of revenue earned by time t
Events will consist of either a customer or an order arriving. The event times are
t_0, the arrival time of the next customer
t_1, the time at which the order being filled will be delivered. If there is, no outstanding order then we take the value of t_1 to be ∞.

The updating is accomplished by considering which of the event times is smaller. If we are presently at time t and we have the values of the preceding variables, then we move along in time as follows.

Case 1: $t_0 < t_1$

Reset: $H = H + (t_0 - t)xh$ since between times t and t_0 we incur a holding cost of $(t_0 - t)h$ for each of the x units in inventory.
Reset: $t = t_0$.
Generate D, a random variable having distribution G. D is the demand of the customer that arrived at time t_0.
Let $w = \min(D, x)$ be the amount of the order that can be filled. The inventory after filling this order is $x - w$.
Reset: $R = R + wr$.
Reset: $x = x - w$.
If $x < s$ and $y = 0$ then reset $y = S - x, t_1 = t + L$.
Generate U and reset $t_0 = t - \frac{1}{\lambda}\log(U)$.

Case 2: $t_1 \leqslant t_0$

Reset: $H = H + (t_1 - t)xh$.
Reset: $t = t_1$.
Reset: $C = C + c(y)$.
Reset: $x = x + y$.
Reset: $y = 0, t_1 = \infty$.

By using the preceding updating schedule it is easy to write a simulation program to analyze the model. We could then run the simulation until the first event occurs after some large preassigned time T, and we could then use $(R - C - H)/T$ as an estimate of the shop's average profit per unit time. Doing this for varying values

of s and S would then enable us to determine a good inventory ordering policy for the shop.

7.6 An Insurance Risk Model

Suppose that the different policyholders of a casualty insurance company generate claims according to independent Poisson processes with a common rate λ, and that each claim amount has distribution F. Suppose also that new customers sign up according to a Poisson process with rate ν, and that each existing policyholder remains with the company for an exponentially distributed time with rate μ. Finally, suppose that each policyholder pays the insurance firm at a fixed rate c per unit time. Starting with n_0 customers and initial capital $a_0 \geqslant 0$, we are interested in using simulation to estimate the probability that the firm's capital is always nonnegative at all times up to time T.

To simulate the preceding, we define the variables and events as follows.

Time Variable t

System State Variable (n, a), where n is the number of policyholders and a is the firm's current capital.

Events There are three types of events: a new policyholder, a lost policyholder, and a claim. The event list consists of a single value, equal to the time at which the next event occurs.

EL t_E

We are able to have the event list consist solely of the time of the next event because of results about exponential random variables that were presented in Section 2.9. Specifically, if (n, a) is the system state at time t then, because the minimum of independent exponential random variables is also exponential, the time at which the next event occurs will equal $t + X$, where X is an exponential random variable with rate $\nu + n\mu + n\lambda$. Moreover, no matter when this next event occurs, it will result from

A new policyholder, with probability $\dfrac{\nu}{\nu + n\mu + n\lambda}$

A lost policyholder, with probability $\dfrac{n\mu}{\nu + n\mu + n\lambda}$

A claim, with probability $\dfrac{n\lambda}{\nu + n\mu + n\lambda}$

After determining when the next event occurs, we generate a random number to determine which of the three possibilities caused the event, and then use this information to determine the new value of the system state variable.

In the following, for given state variable (n, a), X will be an exponential random variable with rate $\nu + n\mu + n\lambda$; J will be a random variable equal to 1 with probability $\dfrac{\nu}{\nu + n\mu + n\lambda}$, to 2 with probability $\dfrac{n\mu}{\nu + n\mu + n\lambda}$, or to

3 with probability $\dfrac{n\lambda}{\nu + n\mu + n\lambda}$; Y will be a random variable having the claim distribution F.

Output Variable I, where

$$I = \begin{cases} 1, & \text{if the firm's capital is nonnegative throughout } [0, t] \\ 0, & \text{otherwise} \end{cases}$$

To simulate the system, we initialize the variables as follows.

Initialize

First initialize

$$t = 0, \quad a = a_0, \quad n = n_0$$

then generate X and initialize

$$t_E = X$$

To update the system we move along to the next event, first checking whether it takes us past time T.

Update Step

Case 1: $t_E > T$:

Set $I = 1$ and end this run.

Case 2: $t_E \leqslant T$:

Reset

$$a = a + nc(t_E - t)$$
$$t = t_E$$

Generate J:

$J = 1$: reset $n = n + 1$
$J = 2$: reset $n = n - 1$
$J = 3$: Generate Y. If $Y > a$, set $I = 0$ and end this run; otherwise reset $a = a - Y$

Generate X: reset $t_E = t + X$

The update step is then continually repeated until a run is completed.

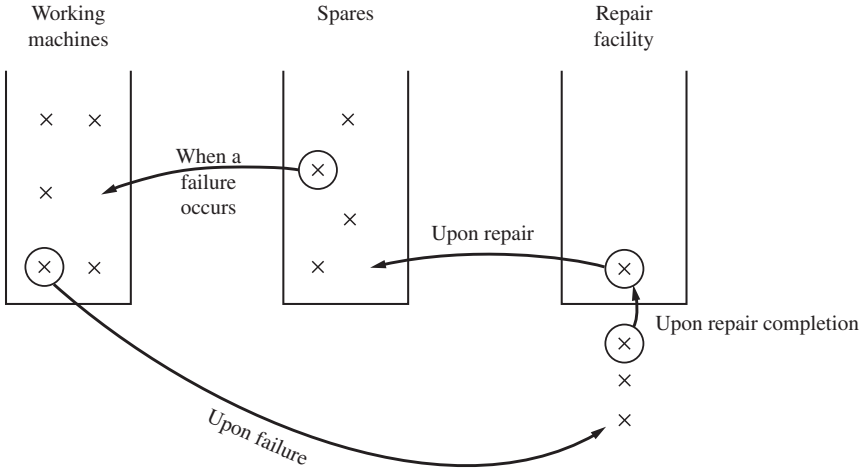

Figure 7.4. Repair Model.

7.7 A Repair Problem

A system needs n working machines to be operational. To guard against machine breakdown, additional machines are kept available as spares. Whenever a machine breaks down it is immediately replaced by a spare and is itself sent to the repair facility, which consists of a single repairperson who repairs failed machines one at a time. Once a failed machine has been repaired it becomes available as a spare to be used when the need arises (see Figure 7.4). All repair times are independent random variables having the common distribution function G. Each time a machine is put into use the amount of time it functions before breaking down is a random variable, independent of the past, having distribution function F.

The system is said to "crash" when a machine fails and no spares are available. Assuming that there are initially $n + s$ functional machines of which n are put in use and s are kept as spares, we are interested in simulating this system so as to approximate $E[T]$, where T is the time at which the system crashes.

To simulate the preceding we utilize the following variables.

Time Variable t

System State Variable r: the number of machines that are down at time t

Since the system state variable will change either when a working machine breaks down or when a repair is completed, we say that an "event" occurs whenever either of these occurs. In order to know when the next event will occur, we need to keep track of the times at which the machines presently in use will fail and the time at

which the machine presently being repaired (if there is a machine in repair) will complete its repair. Because we will always need to determine the smallest of the n failure times, it is convenient to store these n times in an ordered list. Thus it is convenient to let the event list be as follows:

$$\text{Event List} : t_1 \leqslant t_2 \leqslant t_3 \leqslant \cdots \leqslant t_n, t^*$$

where $t_1, \ldots, t_n$ are the times (in order) at which the n machines presently in use will fail, and t^* is the time at which the machine presently in repair will become operational, or if there is no machine presently being repaired then $t^* = \infty$.

To begin the simulation, we initialize these quantities as follows.

Initialize

Set $t = r = 0, t^* = \infty$.
Generate $X_1, \ldots, X_n$, independent random variables each having distribution F.
Order these values and let t_i be the ith smallest one, $i = 1, \ldots, n$.
Set Event list: $t_1, \ldots, t_n, t^*$.

Updating of the system proceeds according to the following two cases.

Case 1: $t_1 < t^*$

Reset: $t = t_1$.
Reset: $r = r + 1$ (because another machine has failed).
If $r = s + 1$, stop this run and collect the data $T = t$ (since, as there are now $s + 1$ machines down, no spares are available).
If $r < s+1$, generate a random variable X having distribution F. This random variable will represent the working time of the spare that will now be put into use. Now reorder the values $t_2, t_3, \ldots, t_n, t+X$ and let t_i be the ith smallest of these values, $i = 1, \ldots, n$.
If $r = 1$, generate a random variable Y having distribution function G and reset $t^* = t + Y$. (This is necessary because in this case the machine that has just failed is the only failed machine and thus repair will immediately begin on it; Y will be its repair time and so its repair will be completed at time $t + Y$.)

Case 2: $t^* \leqslant t_1$

Reset: $t = t^*$.
Reset: $r = r - 1$.
If $r > 0$, generate a random variable Y having distribution function G, and representing the repair time of the machine just entering service, and reset $t^* = t + Y$.
If $r = 0$, set $t^* = \infty$.

The above rules for updating are illustrated in Figure 7.5.

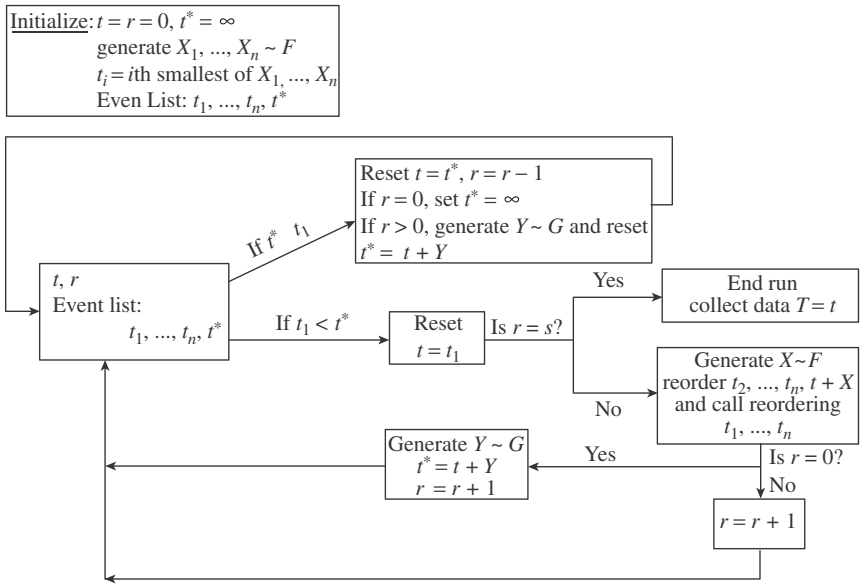

Figure 7.5. Simulating the Repair Model.

Each time we stop (which occurs when $r = s+1$) we say that a run is completed. The output for the run is the value of the crash time T. We then reinitialize and simulate another run. In all, we do a total of, say, k runs with the successive output variables being $T_1, \ldots, T_k$. Since these k random variables are independent and each represents a crash time, their average, $\sum_{i=1}^{k} T_i/k$, is the estimate of $E[T]$, the mean crash time. The question of determining when to stop the simulation—that is, determining the value of k—is considered in Chapter 8, which presents the methods used to statistically analyze the output from simulation runs.

7.8 Exercising a Stock Option

Let $S_n, n \geqslant 0$ denote the price of a specified stock at the end of day n. A common model is to suppose that

$$S_n = S_0 \exp\{X_1 + \cdots + X_n\}, \quad n \geqslant 0$$

where $X_1, X_2, \ldots$ is a sequence of independent normal random variables, each with mean μ and variance σ^2. This model, which supposes that each day's percentage increase in price over the previous day has a common distribution, is called the *lognormal random walk model*. Let $\alpha = \mu + \sigma^2/2$. Suppose now that you own an option to purchase one unit of this stock at a fixed price K, called the *striking*

price, at the end of any of the next N days. If you exercise this option when the stock's price is S then, because you only pay the amount K, we will call this a gain of $S - K$ (since you could theoretically immediately turn around and sell the stock at the price S). The expected gain in owning the option (which clearly would never be exercised if the stock's price does not exceed K during the time period of interest) depends on the option exercising policy you employ. Now, it can be shown that if $\alpha \geqslant 0$ then the optimal policy is to wait until the last possible moment and then exercise the option if the price exceeds K and not exercise otherwise. Since $X_1 + \cdots + X_N$ is a normal random variable with mean $N\mu$ and variance $N\sigma^2$, it is not difficult to explicitly compute the return from this policy. However, it is not at all easy to characterize an optimal, or even a near optimal, policy when $\alpha < 0$, and for any reasonably good policy it is not possible to explicitly evaluate the expected gain. We will now give a policy that can be employed when $\alpha < 0$. This policy, although far from being an optimal policy, appears to be reasonably good. It calls for exercising the option when there are m days to go whenever, for each $i = 1, \ldots, m$, that action leads to a higher expected payoff than letting exactly i days go by and then either exercising (if the price at that point is greater than K) or giving up on ever exercising.

Let $P_m = S_{N-m}$ denote the price of the stock when there are m days to go before the option expires. The policy we suggest is as follows:

Policy: If there are m days to go, then exercise the option at this time if

$$P_m > K$$

and, if for each $i = 1, \ldots, m$

$$P_m > K + P_m e^{i\alpha}\Phi(\sigma\sqrt{i} + b_i) - K\Phi(b_i)$$

where

$$b_i = \frac{i\mu - \log(K/P_m)}{\sigma\sqrt{i}}$$

and where $\Phi(x)$ is the standard normal distribution function and can be accurately approximated by the following formula: For $x \geqslant 0$

$$\Phi(x) \approx 1 - \frac{1}{\sqrt{2\pi}}(a_1 y + a_2 y^2 + a_3 y^3)e^{-x^2/2}$$

For $x < 0$, $\Phi(x) = 1 - \Phi(-x)$; where

$$y = \frac{1}{1 + 0.33267x}$$
$$a_1 = 0.4361836$$
$$a_2 = -0.1201676$$
$$a_3 = 0.9372980$$

Let SP denote the price of the stock when the option is exercised, if it is exercised, and let SP be K if the option is never exercised. To determine the expected worth of the preceding policy—that is, to determine $E[SP] - K$—it is necessary to resort to simulation. For given parameters μ, σ, N, K, S_0 it is easy enough to simulate the price of the stock on separate days by generating X, a normal random variable with mean μ and standard deviation σ, and then using the relation

$$P_{m-1} = P_m e^X$$

Thus, if P_m is the price with m days to go and the policy does not call for exercising the option at this time, then we would generate X and determine the new price P_{m-1} and have the computer check whether the policy calls for exercising at this point. If so, then for that simulation run $SP = P_{m-1}$; if not, then we would determine the price at the end of the next day, and so on. The average value, over a large number of simulation runs, of $SP - K$ would then be our estimate of the expected value of owning the option when you are using the preceding policy.

7.9 Verification of the Simulation Model

The end product of the discrete event approach to simulation is a computer program that one hopes is free of error. To verify that there are indeed no bugs in the program, one should, of course, use all the "standard" techniques of debugging computer programs. However, there are several techniques that are particularly applicable in debugging simulation models, and we now discuss some of them.

As with all large programs one should attempt to debug in "modules" or subroutines. That is, one should attempt to break down the program into small and manageable entities that are logical wholes and then attempt to debug these entities. For example, in simulation models the generation of random variables constitutes one such module, and these modules should be checked separately.

The simulation should always be written broadly with a large number of input variables. Oftentimes by choosing suitable values we can reduce the simulation model to one that can be evaluated analytically or that has been previously extensively studied, so as to compare our simulated results with known answers.

In the testing stage, the program should be written to give as output all the random quantities it generates. By suitably choosing simple special cases, we can then compare the simulated output with the answer worked out by hand. For example, suppose we are simulating the first T time units of a k server queueing system. After inputting the values $T = 8$ (meant to be a small number) and $k = 2$, suppose the simulation program generates the following data:

Customer number:	1	2	3	4	5	6
Arrival time:	1.5	3.6	3.9	5.2	6.4	7.7
Service time:	3.4	2.2	5.1	2.4	3.3	6.2

and suppose that the program gives as output that the average time spent in the system by these six customers is 5.12.

However, by going through the calculations by hand, we see that the first customer spent 3.4 time units in the system; the second spent 2.2 (recall there are two servers); the third arrived at time 3.9, entered service at time 4.9 (when the first customer left), and spent 5.1 time units in service—thus, customer 3 spent a time 6.1 in the system; customer 4 arrived at time 5.2, entered service at time 5.8 (when number 2 departed), and departed after an additional time 2.4—thus, customer 4 spent a time 3.0 in the system; and so on. These calculations are presented below:

Arrival time:	1.5	3.6	3.9	5.2	6.4	7.7
Time when service began:	1.5	3.6	4.9	5.8	8.2	10.0
Departure time:	4.9	5.8	10.0	8.2	11.5	16.2
Time in system:	3.4	2.2	6.1	3.0	5.1	8.5

Hence, the output for the average time spent in the system by all arrivals up to time $T = 8$ should have been

$$\frac{3.4 + 2.2 + 6.1 + 3.0 + 5.1 + 8.5}{6} = 4.71666\ldots$$

thus showing that there is an error in the computer program which gave the output value 5.12.

A useful technique when searching for errors in the computer program is to utilize a *trace*. In a trace, the state variable, the event list, and the counter variables are all printed out after each event occurs. This allows one to follow the simulated system over time so as to determine when it is not performing as intended. (If no errors are apparent when following such a trace, one should then check the calculations relating to the output variables.)

Exercises

1. Write a program to generate the desired output for the model of Section 7.2. Use it to estimate the average time that a customer spends in the system and the average amount of overtime put in by the server, in the case where the arrival process is a Poisson process with rate 10, the service time density is

$$g(x) = 20e^{-40x}(40x)^2, \quad x > 0$$

and $T = 9$. First try 100 runs and then 1000.

2. Suppose in the model of Section 7.2 that we also wanted to obtain information about the amount of idle time a server would experience in a day. Explain how this could be accomplished.

3. Suppose that jobs arrive at a single server queueing system according to a nonhomogeneous Poisson process, whose rate is initially 4 per hour, increases steadily until it hits 19 per hour after 5 hours, and then decreases steadily until it hits 4 per hour after an additional 5 hours. The rate then repeats indefinitely in this fashion—that is, $\lambda(t+10) = \lambda(t)$. Suppose that the service distribution is exponential with rate 25 per hour. Suppose also that whenever the server completes a service and finds no jobs waiting he goes on break for a time that is uniformly distributed on (0, 0.3). If upon returning from his break there are no jobs waiting, then he goes on another break. Use simulation to estimate the expected amount of time that the server is on break in the first 100 hours of operation. Do 500 simulation runs.

4. Fill in the updating scheme for Case 3 in the model of Section 7.4.

5. Consider a single-server queueing model in which customers arrive according to a nonhomogeneous Poisson process. Upon arriving they either enter service if the server is free or else they join the queue. Suppose, however, that each customer will only wait a random amount of time, having distribution F, in queue before leaving the system. Let G denote the service distribution. Define variables and events so as to analyze this model, and give the updating procedures. Suppose we are interested in estimating the average number of lost customers by time T, where a customer that departs before entering service is considered lost.

6. Suppose in Exercise 5 that the arrival process is a Poisson process with rate 5; F is the uniform distribution on (0, 5); and G is an exponential random variable with rate 4. Do 500 simulation runs to estimate the expected number of lost customers by time 100. Assume that customers are served in their order of arrival.

7. Repeat Exercise 6, this time supposing that each time the server completes a service, the next customer to be served is the one who has the earliest queue departure time. That is, if two customers are waiting and one would depart the queue if his service has not yet begun by time t_1 and the other if her service had not yet begun by time t_2, then the former would enter service if $t_1 < t_2$ and the latter otherwise. Do you think this will increase or decrease the average number that depart before entering service?

8. In the model of Section 7.4, suppose that G_1 is the exponential distribution with rate 4 and G_2 is exponential with rate 3. Suppose that the arrivals are according to a Poisson process with rate 6. Write a simulation program to generate data corresponding to the first 1000 arrivals. Use it to estimate

(a) the average time spent in the system by these customers.
(b) the proportion of services performed by server 1.
(c) Do a second simulation of the first 1000 arrivals and use it to answer parts (a) and (b). Compare your answers to the ones previously obtained.

9. Suppose in the two-server parallel model of Section 7.4 that each server has its own queue, and that upon arrival a customer joins the shortest one. An arrival finding both queues at the same size (or finding both servers empty) goes to server 1.

(a) Determine appropriate variables and events to analyze this model and give the updating procedure.

Using the same distributions and parameters as in Exercise 8, find

(b) the average time spent in the system by the first 1000 customers.
(c) the proportion of the first 1000 services performed by server 1.

Before running your program, do you expect your answers in parts (b) and (c) to be larger or smaller than the corresponding answers in Exercise 8?

10. Suppose in Exercise 9 that each arrival is sent to server 1 with probability p, independent of anything else.

(a) Determine appropriate variables and events to analyze this model and give the updating procedure.
(b) Using the parameters of Exercise 9, and taking p equal to your estimate of part (c) of that problem, simulate the system to estimate the quantities defined in part (b) of Exercise 9. Do you expect your answer to be larger or smaller than that obtained in Exercise 9?

11. Suppose that claims are made to an insurance company according to a Poisson process with rate 10 per day. The amount of a claim is a random variable that has an exponential distribution with mean \$1000. The insurance company receives payments continuously in time at a constant rate of \$11,000 per day. Starting with an initial capital of \$25,000, use simulation to estimate the probability that the firm's capital is always positive throughout its first 365 days.

12. Suppose in the model of Section 7.6 that, conditional on the event that the firm's capital goes negative before time T, we are also interested in the time

at which it becomes negative and the amount of the shortfall. Explain how we can use the given simulation methodology to obtain relevant data.

13. For the repair model presented in Section 7.7:

(a) Write a computer program for this model.

(b) Use your program to estimate the mean crash time in the case where $n = 4, s = 3, F(x) = 1 - e^{-x}$, and $G(x) = 1 - e^{-2x}$.

14. In the model of Section 7.7, suppose that the repair facility consists of two servers, each of whom takes a random amount of time having distribution G to service a failed machine. Draw a flow diagram for this system.

15. A system experiences shocks that occur in accordance with a Poisson process having a rate of 1/hour. Each shock has a certain amount of damage associated with it. These damages are assumed to be independent random variables (which are also independent of the times at which the shocks occur), having the common density function

$$f(x) = xe^{-x}, \quad x > 0$$

Damages dissipate in time at an exponential rate α—that is, a shock whose initial damage is x will have remaining damage value $xe^{-\alpha s}$ at time s after it occurs. In addition, the damage values are cumulative. Thus, for example, if by time t there have been a total of two shocks, which originated at times t_1 and t_2 and had initial damages x_1 and x_2, then the total damage at time t is $\sum_{i=1}^{2} x_i e^{-\alpha(t-t_i)}$. The system fails when the total damage exceeds some fixed constant C.

(a) Suppose we are interested in utilizing a simulation study to estimate the mean time at which the system fails. Define the "events" and "variables" of this model and draw a flow diagram indicating how the simulation is to be run.
(b) Write a program that would generate k runs.
(c) Verify your program by comparing output with a by-hand calculation.
(d) With $\alpha = 0.5, C = 5$, and $k = 1000$, run your program and use the output to estimate the expected time until the system fails.

16. Messages arrive at a communications facility in accordance with a Poisson process having a rate of 2/hour. The facility consists of three channels, and an arriving message will either go to a free channel if any of them are free or else will be lost if all channels are busy. The amount of time that a message ties up a channel is a random variable that depends on the weather condition at the time the message arrives. Specifically, if the message arrives when the weather

is "good," then its processing time is a random variable having distribution function

$$F(x) = x, \quad 0 < x < 1$$

whereas if the weather is "bad" when a message arrives, then its processing time has distribution function

$$F(x) = x^3, \quad 0 < x < 1$$

Initially, the weather is good, and it alternates between good and bad periods—with the good periods having fixed lengths of 2 hours and the bad periods having fixed lengths of 1 hour. (Thus, for example, at time 5 the weather changes from good to bad.)
Suppose we are interested in the distribution of the number of lost messages by time $T = 100$.

(a) Define the events and variables that enable us to use the discrete event approach.
(b) Write a flow diagram of the above.
(c) Write a program for the above.
(d) Verify your program by comparing an output with a hand calculation.
(e) Run your program to estimate the mean number of lost messages in the first 100 hours of operation.

17. Estimate, by a simulation study, the expected worth of owning an option to purchase a stock anytime in the next 20 days for a price of 100 if the present price of the stock is 100. Assume the model of Section 7.8, with $\mu = -0.05, \sigma = 0.3$, and employ the strategy presented there.

18. A shop stocks a certain toy. Customers wanting the toy arrive according to a Poisson process with rate λ. Each such customer wants to purchase i of these toys with probability p_i, where $p_1 = \frac{1}{2}$, $p_2 = \frac{1}{3}$, $p_3 = \frac{1}{6}$. The shop initially has 4 such toys, and the owner uses a policy of ordering additional toys only when she has no more toys left. At such times, 10 toys are ordered and immediately delivered. Any customer whose requirements cannot be exactly met departs without making a purchase. (For instance, if there are 2 toys in the shop when a customer wanting 3 arrives then that customer will depart without buying any.) Suppose that we want to use simulation to estimate the expected number of customers who depart without making a purchase in the first T units of time. Show how this can be done using the discrete event approach. Define all variables and show how to update them.

Bibliography

Banks, J., and J. Carson, *Discrete-Event System Simulation*. Prentice-Hall, New Jersey, 1984.

Clymer, J., *Systems Analysis Using Simulation and Markov Models*. Prentice-Hall, New Jersey, 1990.

Gottfried, B., *Elements of Stochastic Process Simulation*. Prentice-Hall, New Jersey, 1984.

Law, A. M., and W. D. Kelton, *Simulation Modelling and Analysis*, 3rd ed. McGraw-Hill, New York, 1997.

Mitrani, I., *Simulation Techniques for Discrete Event Systems*. Cambridge University Press, Cambridge, U.K., 1982.

Peterson, R., and E. Silver, *Decision Systems for Inventory Management and Production Planning*. Wiley, New York, 1979.

Pritsker, A., and C. Pedgen, *Introduction to Simulation and SLAM*. Halsted Press, New York, 1979.

Shannon, R. E., *Systems Simulation: The Art and Science*. Prentice-Hall, New Jersey, 1975.

Solomon, S. L., *Simulation of Waiting Line Systems*. Prentice-Hall, New Jersey, 1983.