Chapter 10
Discrete-Event Simulation

David Goldsman and Paul Goldsman

10.1 Introduction

A *discrete-event simulation* (DES) is characterized by changes in the simulation's state at discrete time points. Examples of systems that might be evaluated using DES include:

- Any queueing system, such as a bank service counter, where customers arrive occasionally, wait in lines for service, eventually receive service, and depart.
- Manufacturing systems, where parts are processed in various sequences at different stations, after which they leave the plant.
- Inventory systems, where random quantities of a certain product are purchased by customers each day at a store, and where resupplies of the product move from stage to stage of the supply chain before being purchased at the store.

The above examples are driven by various events that occur at discrete times and change the state of the simulation. Event times correspond to, for example, customer arrivals, customer departures from a server or from the system, machine breakdowns, and even an "end-of-simulation" event. DES contrasts with the so-called *continuous-time simulation modeling*, in both the nature of the systems studied and the methods by which the systems are analyzed. Examples of continuous-time models include:

- Weather systems and other natural phenomena
- Electronic circuit systems
- Vehicular traffic movement

D. Goldsman (✉)
Industrial and Systems Engineering, Georgia Institute of Technology,
Atlanta, GA, USA
e-mail: sman@gatech.edu

P. Goldsman
Syracuse, NY, USA
e-mail: pgoldsman@gmail.com

M. L. Loper (ed.), *Modeling and Simulation in the Systems Engineering Life Cycle,*
Simulation Foundations, Methods and Applications,
DOI 10.1007/978-1-4471-5634-5_10

Continuous-time systems are often modeled via sets of differential equations. In this chapter, we will give a high-level description of how a DES works, followed by a running example illustrating the salient concepts, and then a brief discussion of specialty DES software.

10.2 How Does a DES Work?

Discrete-event simulations almost always maintain what is known as a *future events list* (FEL), which keeps track of the set of upcoming events. The simulation evolves over time by proceeding from event to event on the FEL, which is ordered by the simulation clock time. Each time an event occurs, we update the state of the simulation, any appropriate statistics, and finally the FEL. Then we go to the next event on the updated FEL. The FEL is usually handled as a linked list, so insertions and deletions of events on the list are easy to carry out.

For example, consider a trivial FEL that is initialized at the start of a simple queueing simulation. We will assume that customers arrive to be served by a single server, in a first-in-first-out (FIFO) manner. The random variables corresponding to customer arrival and service times are generated by well-known algorithms (see, for instance, Law 2014).

Suppose that we have the following situation at time 0:

- System state and FEL initialization—the system starts out with no one in the system. Our minimalist FEL consists of two events: (i) start the simulation at time 0 and (ii) generate the first customer arrival at time 3.
- Statistics initialization—we create a statistic to keep track of the average number of customers in the system (initialized to 0).
- FEL update—at time 0, the simulation starts, and the start event is removed from the FEL.

The simulation clock then jumps to the next event (and in this case the only event) on the FEL: the first customer's arrival at time 3. At this point:

- System state update—the first customer starts getting served by the single server.
- Statistics update—the average number of customers in the system up to time 3 is still 0.
- FEL update—delete the arrival at time 3 from the FEL, but we now can add two more events to the FEL: the next customer arrival at time 7, and the service completion of the first customer, scheduled to occur at time 11.

The simulation clock then jumps to the next event at time 7, and the cycle of tasks repeats itself. This sequence of events continues until the end-of-simulation event occurs.

10.3 Queueing Example

Perhaps the nature of a DES can be best understood by way of a simple example whose evolution can actually be carried out by hand. To this end, suppose that we are interested in evaluating a single-server queueing system. Our system in this case is a fast-food restaurant, where customers arrive one at a time, according to some stochastic (i.e., random) process. Typically, the times between arrivals are independent and identically distributed (IID; for instance, from a Poisson process), though violations of this IID assumption can easily be accommodated in practice. In any case, upon arrival, customers line up in a FIFO queue and are eventually processed by the server. Service times are also stochastic and often IID from some service–time distribution.

Thus, we will create customers who:

- Enter the restaurant
- Wait in the line (if there is one) in front of the single server
- Order their food when their turn comes
- Leave after their food is prepared by the server

Some obvious goals of our DES are to estimate:

- Expected customer waiting times
- Expected number of people in the system (in line and being served)
- Percentage of time that the server is busy

Customer waiting times are important measures related to customer satisfaction—customers are in a hurry at a fast-food restaurant, and simply do not like to wait. Management also does not like long lines—in addition to annoying the customers, long lines take up limited space that could be better utilized. One way to reduce queue length is to hire more workers, but this can get expensive. So management needs to look simultaneously at customer waiting times, queue lengths, and server utilizations, in order to strike some sort of synergistic balance between pleasing the customers and getting the most out of its workers.

We can illustrate these performance criteria via a hand simulation. Let us suppose that there are a total of six customers who show up starting at 4:00 p.m. We will start by defining some useful notation:

- i = the customer number, by order of arrival ($i = 1,\ldots,6$)
- A_i = customer i's arrival time
- S_i = customer i's service time

Suppose the customer arrival times (A_i) and service times (S_i) shown in Table 10.1 are, for notational convenience, all given in whole minutes. In order to carry out the hand simulation, we will also assume that no other customers are in the system when customer 1 arrives (so he will not have to wait in line), and that no one besides our six customers arrives in the system until after customer 6 departs. Notice that this system has *two* sources of randomness: the arrival times and the service times.

Table 10.1 Customer arrival and service times

i	A_i	S_i
1	4:03	7
2	4:04	6
3	4:06	4
4	4:10	6
5	4:15	1
6	4:20	2

Lastly, we need to define a little more notation that will be useful for bookkeeping purposes:

- T_i = the time that customer i starts service
- $W_i = T_i - A_i$ = the amount of time customer i has to wait in line before getting served
- $D_i = T_i + S_i$ = customer i's departure time

In general, if customer i shows up and nobody else is in the system, then he is served immediately (i.e., $T_i = A_i$). Otherwise, the system is not empty when customer i arrives, and he must wait until customer i-1 (the customer just ahead of him) completes service (i.e., $T_i = D_i$). Thus, we immediately see that

$$T_i = max(A_i, D_i), \quad i = 1, \ldots, 6.$$

With this piece of the puzzle in hand, we can come up with Table 10.2, which tells the entire story of customer arrivals and departures.

Customer 1's arrival event is at time A_1 = 4:03 p.m.; he does not have to wait in line, since he is the only customer in the system. He uses the server for $S_1 = 7$ min and therefore his departure event occurs at time D_i = 4:10 p.m. Customer 2 shows up at time A_2 = 4:04 p.m. and finds that she must wait until time $T_2 = max(A_2, D_1) =$ 4:10 p.m. to begin service. Thus, she has to wait $W_2 = 6$ min. Her service time is $S_2 = 6$ min, so she will leave at time D_2 = 4:16 p.m. This exercise continues until the sixth customer departs at time 4:29 p.m., which is the end of the simulation.

Table 10.2 Summary of customer arrivals and departures

i	A_i	T_i	W_i	S_i	D_i
1	4:03	4:03	0	7	4:10
2	4:04	4:10	6	6	4:16
3	4:06	4:16	10	4	4:20
4	4:10	4:20	10	6	4:26
5	4:15	4:26	11	1	4:27
6	4:20	4:27	7	2	4:29

Table 10.3 How events affect the number of customers in the system

Time t	Event	$L(t)$
4:00	Simulation begins	0
4:03	Customer 1 arrives	1
4:04	Customer 2 arrives	2
4:06	Customer 3 arrives	3
4:10	Cust. 1 departs; cust. 4 arrives	3
4:15	Customer 5 arrives	4
4:16	Customer 2 departs	3
4:20	Cust. 3 departs; cust. 6 arrives	3
4:26	Customer 4 departs	2
4:27	Customer 5 departs	1
4:29	Cust. 6 departs; simulation ends	0

Note that we can produce the complete hand simulation table; let us turn our attention to the question of studying the performance of the restaurant. For now, we are primarily interested in customer waiting times and the number of customers in the system at any given time.

The calculation of the average waiting time for the six customers is trivial:

$$W_{avg} = (W_1 + W_2 + \cdots + W_6)/6 = 7.33 \text{ min.}$$

What remains is to calculate the average number of customers in the system; this includes those in line as well as those being served over the time period 4:00–4:29 (the time when customer 6 departs).

Let $L(t)$ denote the number of people in the system at time t. Note that $L(t)$ can only change when customers arrive or depart, i.e., at the discrete event times. Table 10.3 shows the various events that can result in a change in $L(t)$.

The table shows that $L(t)=0$ between times 4:00 and 4:03 p.m., when customer 1 arrives. $L(t)=1$ between times 4:03 and 4:04 p.m., because he is the only customer in the system. Customer 2 arrives at time 4:04 p.m., and, between times 4:04 and 4:06 p.m., there are two customers in the system—customer 1, who is still being served, and customer 2, who now has to wait in line. At time 4:06 p.m., customer 3 arrives, so that there are now three people in the system (two in line and one being served). At time 4:10 p.m., customer 1 departs, while customer 4 simultaneously arrives (so $L(t)$ remains at 3). And so on. The arrival and departure events continue until time 4:29 p.m., when the last customer (customer 6) leaves the system. Fig. 10.1 represents these details pictorially.

Using either Table 10.3 or Fig. 10.1, we see that:

$L(t)=0$, from times $t=$4:00 to $t=$4:03 (a total of 3 min)

$L(t)=1$, from 4:03 to 4:04, and from 4:27 to 4:29 (3 min)

$L(t)=2$, from 4:04 to 4:06, and from 4:26 to 4:27 (3 min)

$L(t)=3$, from 4:06 to 4:15, and from 4:16 to 4:26 (19 min)

$L(t)=4$, from 4:15 to 4:16 (1 min)

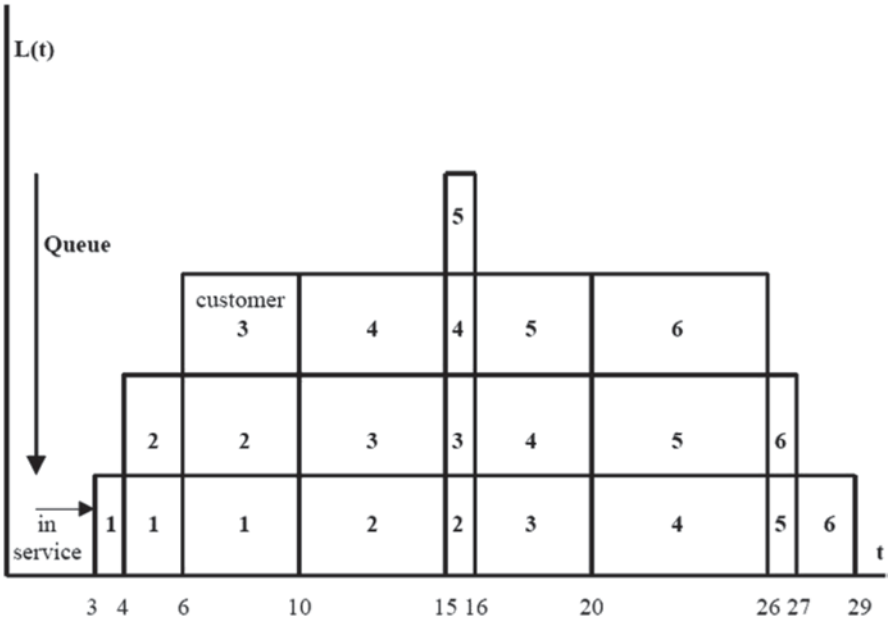

Fig. 10.1 The first six customers at the restaurant

Let T_i represent the amount of time during the interval t=4:00 to t=4:29 in which there are exactly i customers in the system. From the above calculations, we have:

$$T_0 = 3,\ T_1 = 3,\ \ T_2 = 3,\ T_3 = 19,\ \textit{and}\ \ T_4 = 1.$$

Thus, there are exactly zero customers for precisely 3 min, one customer in the system for 3 min, two customers for 3 min, etc. The average value of $L(t)$ is now simply:

$$L_{avg} = [0T_0 + 1T_1 + 2T_2 + 3T_3 + 4T_4]/29 = 2.41.$$

Therefore, on average, there were 2.41 people in the system (in the waiting line or being served) between 4:00 and 4:29 p.m.

Figure 10.1 can also be used to determine the proportion of time that the server is being used. Since customers are in service continuously from 4:03 p.m. to the end of the simulation at 4:29 p.m., we immediately see that the server is busy 26/29=89.7% of the time.

10.4 Generalizations and Beyond

The simple hand simulation that we conducted only hints at what lies under the surface of an actual simulation language. Such languages can handle multiple streams of customer arrivals into large queueing networks, with complicated waiting-line behavior (e.g., balking when a line is perceived to be too large, jockeying between lines, etc.) and service disciplines (e.g., last-in-first-out services, service by priority, etc.). These languages all maintain an FEL that can process almost any set of events.

For further information on DES, one should consult the standard references: (Banks et al. 2010; undergraduate level), (Law 2014; masters level), and (Nelson 2013; advanced). In addition, the yearly *Proceedings of the Winter Simulation Conference,* archived at www.wintersim.org, contain a wealth of articles on DES, including relevant tutorials on DES theory, applications, and languages. With regard to the many DES languages on the market, the reader is advised to consult James Swain's *Simulation Software Survey,* which appears periodically in OR/MS *Today,* the trade magazine of The Institute for Operations Research and the Management Sciences (INFORMS; www.informs.org).

Acknowledgments The first author was partially supported by National Science Foundation grants CMMI-0927592 and CMMI-1233141.

References

Banks J, Carson JS, Nelson BL, Nicol DM (2010) Discrete-event system simulation, 5th edn. Prentice Hall, Englewood Cliffs

Law AM (2014) Simulation modeling and analysis, 5th edn. McGraw-Hill, New York

Nelson BL (2013) Foundations and methods of stochastic simulation: a first course. Springer-Verlag, New York