Discrete Event Simulation

Alireza Sheikh-Zadeh

What is discrete event simulation?

Discrete event simulation is an important tool for the modeling of complex queuing systems. It represents manufacturing, transportation, and service systems in a computer program to perform and test different planning experiments (e.g., does the urgent care needs one front desk staff or two?)

Discrete event simulation refers to the simulation of systems that have abrupt, i.e., discrete, changes. For instance, when a new job arrives in a queuing system, the queue length abruptly increases by 1. Simulation of continuous systems such as weather systems or power systems, on the other hand, would not fit this definition, as quantities such as temperature vary continuously.

Why we need simulation

Imagine we try to analyze the following situation. Patients arrive at an emergency room. The arrival of the patients to the emergency department occurs randomly and may vary with the day of the week and even the day's hours. The hospital has a triage station where the arriving patient's condition is monitored. If the patient's condition warrants immediate attention, the patient is expedited to an emergency room bed attended by a doctor and a nurse. In this case, the patient's admitting information may be obtained from a relative. If the patient does not require immediate attention, the patient goes through the admitting process, where the patient's information is received. The patient is then directed to the waiting room to wait for allocation to a room, a doctor, and a nurse. The doctors and nurses within the emergency department must monitor the patients' health by performing tests and diagnosing their symptoms. This process occurs periodically. As the patient receives care, the patient may be moved to and require other facilities (e.g., magnetic resonance imaging (MRI), X-ray, etc.). Eventually, the patient is either discharged after receiving care or admitted to the main hospital. The hospital is interested in conducting a study of the emergency department to improve patients' care while better utilizing the available resources. To investigate this situation, you might need to understand the behavior of specific measures of performance:

- The average number of patients who are waiting.
- The average waiting time of the patients and their average total time in the emergency department.
- The average number of rooms required per hour.

- The average utilization of the doctors and nurses (and other equipment).

Because of the importance of emergency department operations, the hospital has historical records available on the department's process through its patient tracking system. With these records, you might be able to estimate the current performance of the emergency department. Despite this information's availability, when conducting a study of the emergency department, you might want to propose changes to how the department will operate (e.g., staffing levels) in the future. Thus, you are faced with trying to predict the system's future behavior and its performance when making changes to the system. You cannot realistically experiment with the existing system without possibly endangering the patients' lives or care in this situation. Thus, it would be better to model the system and test the effect of the model changes. If the model has acceptable fidelity, you can infer how the changes will affect the existing system. This is where simulation techniques can be utilized.

A key advantage of simulation modeling is that it can model the entire system and its complex interrelationships. The suggestive power of simulation provides flexible modeling that is required for capturing complex processes. As a result, all the system's different components' critical interactions can be accounted for within the model. The modeling of these interactions is inherent in simulation modeling because simulation imitates the real system's behavior (as closely as necessary). The prediction of the system's future behavior is then achieved by monitoring the behavior of different modeling scenarios as a function of simulated time. Real-world systems are often too complex for analytical models and often too expensive to experiment with directly. Simulation models allow modeling this complexity and enable low-cost experimentation to make inferences about how the existing system might behave.

How the discrete-event cock works

In DES, an **event** is something that happens at an instant in time that corresponds to a change in system **state**. An event can be conceptualized as a transmission of information that causes an action resulting in a system state change. Let us consider a simple bank having two tellers that serve customers from a single waiting line. In this situation, the interest system is the bank tellers (whether they are idle or busy) and any customers waiting in the queue. Assume that the bank opens at 9 am, which can be used to designate time zero for the simulation. It should be clear that if the bank does not have any customers arrive during the day, the bank's state will not change. Besides, when a customer arrives, the number of customers in the bank will increase by one. In other words, an arrival of a customer will change the state of the bank. Thus, the arrival of a customer can be considered an event.

Suppose the first customer arrives at time 2 (9:02 am) and there is an elapse of 3 min before the second customer arrives at time 5 (9:02 am). From the discrete-event perspective, nothing is happening in the bank from time [0,2); however, at time 2, an arrival event occurs. The subsequent actions associated with this event need to be

accounted for concerning the state of the system. What are the activities that occur when a customer arrives at the bank?

- The customer enters the waiting line.
- If there is an available teller, the customer will immediately exit the line, and the available teller will begin to provide service.
- If there are no tellers available, the customer will remain to wait in the line until a teller becomes available.

Key components of DES

Entity

The entity is an object of interest in the system whose movement or operation within the system may cause the occurrence of events. Entities can interact with each other. For example, they may compete for a resource, e.g., two patients compete for a doctor as a resource in an urgent care clinic.

Example: a person, car, or train.

Attributes

Attributes are variables that pertain to an individual entity, and they are carried along with the entity, and they go through the system.

Example: Age, gender, amount of money.

Resource

A limited quantity of items that are used (e.g., seized and released) by entities as they proceed through the system. A resource has a capacity that governs the total quantity of items that may be available. All the items in the resource are homogeneous, meaning that they are indistinguishable. If an entity attempts to seize a resource that does not have any units available, it must wait in a queue.

Example: ATM, a doctor, a bank with three tellers.

One of the measurable outcomes of DES is resource utilization.

Resource utilization = (resource busy time)/(total time)

Queue

Queue is a location that holds entities when their movement is constrained within the system.

Queue might have one of the following logics:

- FIFO (first-in, first-out)
- LIFO (last-in, first-out)
- HVF (high value first)

Events

An instantaneous occurrence or action that changes the state of the system at a particular point in time.

Example: the entity has arrived; the entity has left the system.

R Simmer

This class uses Simmer as a process-oriented and trajectory-based Discrete-Event Simulation (DES) package in R.

Example: clinic

Let's say we want to simulate a Clinic where a patient is first seen by a nurse for an intake, next by a doctor for the consultation, and finally by administrative staff to check out or schedule a follow-up appointment.

- The clinic opens from 8 am to 5 pm (540 minutes)
- Nurse service time is normally distributed: Normal(mean = 15, sd = 1) minutes
- Doctor service time is normally distributed: Normal(mean = 20, sd = 1) minutes
- Administrator service time is normally distributed: Normal(mean = 5, sd = 1) minutes
- Patients inter-arrival time is normally distributed: Normal(mean = 5, sd = 0.5) minutes

```
# install.packages("simmer")

library(simmer)

set.seed(123)

# instantiate a new simulation environment
env <- simmer("Clinic")
env

## simmer environment: Clinic | now: 0 | next:
## { Monitor: in memory }
```

```
# Set up a simple trajectory for a patient
patient <- trajectory("patients' path") %>%

  ##patient seizes one nurse
  seize("nurse", 1) %>%
  timeout(function() rnorm(1, 15, 1)) %>%
  release("nurse", 1) %>%

  seize("doctor", 1) %>%
  timeout(function() rnorm(1, 20, 1)) %>%
  release("doctor", 1) %>%

  seize("administration", 1) %>%
  timeout(function() rnorm(1, 5, 1)) %>%
  release("administration", 1)

library(simmer.plot)

## Loading required package: ggplot2

##
## Attaching package: 'simmer.plot'

## The following objects are masked from 'package:simmer':
##
##     get_mon_arrivals, get_mon_attributes, get_mon_resources

plot(patient)

# or plot(patient, verbose = T)
```

In this case, the timeout activity's argument is a function that is evaluated dynamically to produce a stochastic waiting time, but it could be a constant. Apart from that, this function may be as complex as you need and may do whatever you want: interact with your simulation model entities, get resources' status, and make decisions according to the results.

Once the trajectory is known, you may attach arrivals to it and define the resources needed. In the example below, three types of resources are added: the *nurse* and *administration* resources, each one with a capacity of 2, and the *doctor* resource, with a capacity of 3.

The last method adds a generator of arrivals (patients) following the trajectory `patient`. The time between patients is about 5 minutes (a Gaussian of `mean=5` and `sd=0.5`). (Note: returning a negative interarrival time at some point would stop the generator).

```
env %>%

  add_resource("nurse", 2) %>%

  add_resource("doctor", 3) %>%
```

```
  add_resource("administration", 2) %>%

  add_generator("patient", patient, function() rnorm(1, 5, 0.5))
```

```
## simmer environment: Clinic | now: 0 | next: 0
## { Monitor: in memory }
## { Resource: nurse | monitored: TRUE | server status: 0(2) | queue status:
0(Inf) }
## { Resource: doctor | monitored: TRUE | server status: 0(3) | queue status:
0(Inf) }
## { Resource: administration | monitored: TRUE | server status: 0(2) | queue
status: 0(Inf) }
## { Source: patient | monitored: 1 | n_generated: 0 }
```

The simulation is now ready for a test run.

```
env %>%
  run(540)
```

```
## simmer environment: Clinic | now: 540 | next: 541.290201939829
## { Monitor: in memory }
## { Resource: nurse | monitored: TRUE | server status: 2(2) | queue status:
35(Inf) }
## { Resource: doctor | monitored: TRUE | server status: 3(3) | queue status:
0(Inf) }
## { Resource: administration | monitored: TRUE | server status: 1(2) | queue
status: 0(Inf) }
## { Source: patient | monitored: 1 | n_generated: 108 }
```

```
# look under the hood
# head(get_mon_arrivals(env))
# head(get_mon_arrivals(env, per_resource = T))
# head(get_mon_resources(env))
```

State of the system at the end of the day:

- Both nurses are busy, and 35 people are in the queue for nurse
- 3 of the three doctors are busy, and 0 people are in the line for the doctor
- 1 of the two administrators are occupied, and 0 people are in the queue for administrator

What is your suggestion to improve the efficiency of the system?