UNIT 5. GENERAL PRINCIPLES, SIMULATION SOFTWARE

Development of the common framework for the modeling of complex systems by using discrete event simulation is discussed here. The basic building blocks of all discrete event simulation models are entities, attributes, activities and events. In discrete-even simulation, a system is modeled in terms of

- Its state at each point in time
- Entities that pass through the system
- Entities that represents system resources
- Activities and events that cause the system state to change

Discrete event models are appropriate for the system in which system-state changes only at discrete points in time.

5.1 CONCEPTION OF DISCRETE-EVENT SIMULATION

Major concepts of discrete event simulation are defined here –

- **System:** A collection of entities like people, machines etc. that interact together over time to accomplish or more goals.
- **Model:** An abstract representation of a system, usually containing structural, logical or mathematical relationships that describe a system in terms of state, entities and their attributes, sets, processes, events, activities and delays.
- **System State:** A collection of variables that contain all the information necessary to describe the system at any time.
- **Entity:** Any object or component in the system that requires explicit representation in the model like a server, a customer, a machine etc.
- **Attributes:** The properties of a given entity like the priority of a waiting customer, the routing of a job through a job shop etc.
- **List:** A collection of associated entities ordered in some logical fashion (such as all customers currently in a waiting line, ordered by first come, first served, or by priority).
- **Event:** An instantaneous occurrence that changes the state of a system as an arrival of a new customer.
- **Event notice:** A record of an event to occur at the current or some future time, along with any associated data necessary to execute the event; at a minimum, the record includes the event type and the event time.
- **Event list:** A list of event notices for future events, ordered by time of occurrence; also known as the future event list (FEL).
- **Activity:** A duration of time of specified length (e.g., a service time or arrival time), which is known when it begins.
- **Delay:** A duration of time of unspecified indefinite length, which is not known until it ends (Example: a customer's delay in a last-in, first-out waiting line which, when it begins, depends on future arrivals).
- **Clock:** A variable representing simulated time, called as CLOCK.

Example: Consider the Able-Baker call center system discussed in the previous chapter. A discrete-event model has the following components:

System State

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

$L_A(t)$, takes the values 0 or 1 to indicate Able is idle or busy at the time t

$L_B(t)$, takes the value 0 or 1 to indicate Baker is idle or busy at the time t

Entities Neither the callers nor the servers need to be explicitly represented, except in terms of the state variables, unless certain caller averages are desired.

Events

Arrival event
Service completion by Able
Service completion by Baker

Activities

Inter-arrival time
Service time of Able
Service time of Baker

Delay

A caller's wait in queue until Able or Baker becomes free.

5.1.1 The Event Scheduling/Time-Advance Algorithms

The mechanism for advancing simulation time and guaranteeing that all events occur in correct chronological order is based on the future event list (FEL). This list contains all event notices for events that have been scheduled to occur at a future time.

At any given time t, the FEL contains all previously scheduled future events and their associated event times t1, t2, t3,… as shown below. The events in FEL are arranged chronologically – that is,

$$t < t1 \le t2 \le t3 \le \ldots. \le tn$$

CLOCK	System state	Entities and Attributes	Set1	Set2	…….	Future event list (FEL)	Cumulative Statistics and counters
t	(x, y, z, …)					(3, t1) – Type 3 event to occur at time t1 (1, t2) – Type 1 event to occur at time t2 ……………..	

Time *t* is the current value of simulated time, CLOCK. The event associated with the time *t1* is the next event that will occur, and is called as the ***imminent event.*** After the system snapshot at CLOCK = t has been updated, the CLOCK is moved to CLOCK = t1. And, the imminent event notice is removed from the FEL and is executed. The execution of imminent event means that a new system snapshot for the time t1 is created. The sequence of actions that a simulator must perform to advance the clock and build a new system snapshot is called as ***event – scheduling/time – advance algorithm***. The steps of this algorithm are as listed below –

Old system snapshot at time t:

CLOCK	System state	……..	Future event list	………
t	(5, 1, 6)		(3, t1) – Type 3 event to occur at time t1 (1, t2) – Type 1 event to occur at time t2 (1, t3) – Type 1 event to occur at time t3 ---------------------------------- ------------------------------------ (2, tn) – Type 2 event to occur at time tn	

Event – Scheduling / Time – advance Algorithm

Step 1. Remove the event notice for the imminent event (event 3, time t1) from FEL
Step 2. Advance CLOCK to imminent event time (i.e. advance CLOCK from t to t1).
Step 3. Execute imminent event: update system state, change entity attributes and set membership as needed.
Step 4. Generate future events (if necessary) and place their event notices on FEL, ranked by event time. (Example: Event 4 to occur at time t*, where t2 < t* < t3)
Step 5. Update cumulative statistics and counters.

New system snapshot at time t1:

CLOCK	System state	……..	Future event list	………
t1	(5, 1, 5)		(1, t2) – Type 1 event to occur at time t2 (4, t*) – Type 4 event to occur at time t* (1, t3) – Type 1 event to occur at time t3 ------------------------------------ ------------------------------------	

5.1.2 World Views

While using a simulation package, a modeler adopts a world view or orientation for developing a model. The most common world views are –

- **Event scheduling world view:** Here, a simulation analyst concentrates on events and their effect on system state.
- **Process – interaction world view:** The analyst defines the simulation model in terms of entities or objects and their life cycle as they flow through the system, demanding resources and queuing to wait for resources. That is, a process is the life cycle of one entity, and the life cycle consists of various events and activities.
- **Activity – scanning world view:** This uses a fixed time increment and a rule – based approach to decide whether any activities can begin at each point in

simulated time. The activity – scanning approach is later modified to involve certain advantages of event – scheduling approach. Now, it allows for variable time advance and the avoidance of scanning when it is not necessary. This is called as three-phase approach for activity scanning. Here, events are considered to be activities of duration zero time units and hence, they are of two categories:

- o **B Activities –** activities bound to occur. All primary events and conditional activities fall into this category.
- o **C Activities –** activities/events that may happen only upon certain conditions.

With the 3 phase approach, the simulation is carried out with repeated execution till it is completed as below:

- **Phase A** Remove the imminent event from the FEL and advance the clock to its event time. Remove any other events that have the same event time from the FEL.
- **Phase B** Execute all B-type events that were removed from the FEL.
- **Phase C** Scan the conditions that trigger each C type activity and activate any other activity whose conditions are met. Rescan until no additional C type activities can begin and no events occur.

5.1.3 Manual Simulation Using Event Scheduling

Here, a simulation table is used to record the successive system snapshots as the time passes. This is illustrated using an example.

Example: Consider the grocery store with one counter illustrated in Chapter 4 (Singe – server queuing example). This system consists of customer waiting in a queue and the one who is getting service (if any) at the counter. A stopping time of 60 minutes is set of this example. The inter arrival time and service time distributions are given below –

Arrival Time in Mins (1)	**Probability (2)**
1	0.125
2	0.125
3	0.125
4	0.125
5	0.125
6	0.125
7	0.125
8	0.125

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

The random numbers for inter-arrival time are: 913, 727, 015, 948, 309, 922, 753, 235, 302. The random numbers for service time are: 84, 10, 74, 53, 17, 79, 91, 67, 89, 38. The computation of inter-arrival time and service time are as given in Unit 4 example (page numbers 4 – 6).

The model has following components:

System State

LQ(t) – the number of customers in the waiting line
LS(t) – the number being server (0 or 1) at time t

Entities The server and customers are not explicitly modeled except in terms of state variable.

Events

Arrival (A)
Departure (D)
Stopping event (E), scheduled to occur at time 60

Event notices

(A, t) represents that an arrival event to occur at future time *t.*
(D, t) represents a customer departure at future time *t*
(E, 60) represents the simulation stoop event at future time 60

Activities

Inter-arrival time (defined in
Service Time

Delay Customer time spent in waiting line.

The event notices are written as ordered pair – (event type, event time). In this model, FEL will always contain 2-3 event notices. The effect of arrival and departure events are as shown in Figure 5.1 and Figure 5.2.

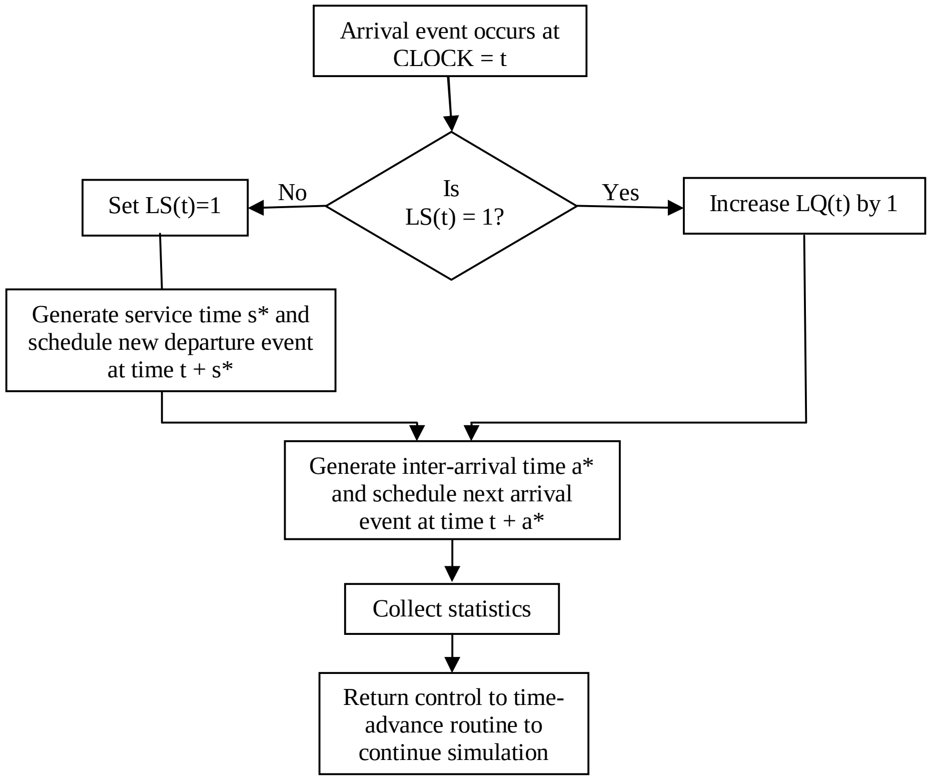
Arrival event occurs at CLOCK = t
Is LS(t) = 1?
No
Yes
Set LS(t)=1
Increase LQ(t) by 1
Generate service time s* and schedule new departure event at time t + s*
Generate inter-arrival time a* and schedule next arrival event at time t + a*
Collect statistics
Return control to time-advance routine to continue simulation

Figure 5.1 Execution of the arrival event

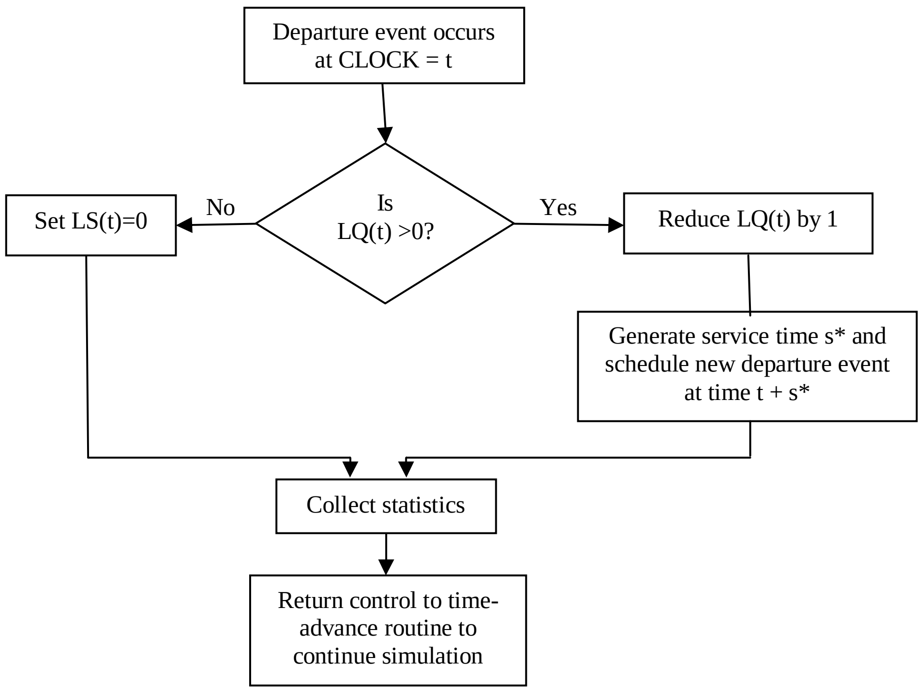
Departure event occurs at CLOCK = t
Is LQ(t) >0?
No
Yes
Set LS(t)=0
Reduce LQ(t) by 1
Generate service time s* and schedule new departure event at time t + s*
Collect statistics
Return control to time-advance routine to continue simulation

Figure 5.2 Execution of the departure event

With the help of given random numbers and the probability distribution table, the inter-arrival time and service time of the customers have been computed as shown below (procedure in explained in Unit 4, page 4-6).

Customer (1)	**Inter-arrival time** (2)	**Arrival time in clock** (3)	**Service time** (4)
1	0	0	4
2	8	8	1
3	6	14	4
4	1	15	3
5	8	23	2
6	3	26	4
7	8	34	5
8	7	41	4
9	2	43	5
10	3	46	3
Total	**46**		**35**

The simulation table for checkout counter is as shown in Table 5.3 and it is generated as explained below.

i. First customer arrives at 0 and starts getting service. Now, the number of customers waiting in a line, LQ(t) = 0 and the number being served, LS(t)=1. Also, note that the second customer arrives at the time 8 and the service time for first customer is 4. Hence, the departure event (of first customer) will occur first and then the arrival event (of second customer) will occur. This is indicated in FEL.

ii. The notations a* is arrival time and s* is the service time.
iii. Server busy time is denoted by B and maximum queue length is denoted by MQ.
iv. Arrival and departures are noted in clock and accordingly LQ(t) and LS(t) are written in the table.
v. The comment column will give the explanation about the arrival and departure of events along with the respective times (a* and s*).
vi. B and MQ are computed accordingly.

Table 5.3 Simulation table for checkout counter

CLOCK	**System State**		**Future Event List**	**Comment**	**Cumulative Statistics**	
	LQ(t)	**LS(t)**			**B**	**MQ**
0	0	1	(D, 4), (A, 8), (E, 60)	First A occurs at 0. Schedule next A (a*=8)and next D (s*=4)	0	0
4	0	0	(A, 8), (E, 60)	1st D occurs	4	0
8	0	1	(D, 9), (A, 14), (E, 60)	2nd A occurs. Schedule next A (a*=14) and next D (s*=1)	4	0
9	0	0	(A, 14), (E, 60)	2nd D occurs	5	0
14	0	1	(A,15), (D, 18), (E, 60)	3rd A occurs. Schedule next A (a*=15) and next D (s*=4)	5	0
15	1	1	(D, 18), (A, 23), (E, 60)	4th A occurs, customer delayed, next a*=8	5	1
18	0	1	(D, 21), (A, 23), (E, 60)	3rd D occurs, schedule next s*=3	9	0
21	0	0	(A, 23), (E, 60)	4th D occurs	12	0
23	0	1	(D, 25), (A, 26), (E, 60)	5th A occurs, schedule next A (a*=3) and next D (s*=2)	12	0
25	0	0	(A, 26), (E, 60)	5th D occurs	14	0
26	0	1	(D, 30), (A, 34), (E, 60)	6th A occurs, schedule next A (a*=8) and next D (s*=4)	14	0
30	0	0	(A, 34), (E, 60)	6th D occurs	18	0
34	0	1	(D, 39), (A, 41), (E, 60)	7th A occurs, schedule next A (a*=7), next D (s*=5)	18	0
39	0	0	(A, 41), (E, 60)	7th D occurs	23	0
41	0	1	(A, 43), (D, 45), (E, 60)	8th A occurs, schedule next A (a*=2) and next D (s*=4)	23	0
43	1	1	(D, 45), (A, 46), (E, 60)	9th A occurs, customer delayed, next a*=3	23	1
45	0	1	(A, 46), (D, 50), (E, 60)	8th D occurs, schedule next A (a*=1), next D(s*=5)	27	0
46	1	1	(D, 50), (E, 60)	10th A occurs. Schedule next D (s*=3)	27	1
50	0	1	(D, 53), (E, 60)	9th D occurs	32	0
53	0	0	(E, 60)	10th D occurs	35	0

5.2 LIST PROCESSING

List processing deals with the methods for handling lists of entities and the future event list. Simulation packages provide facilities for an analyst to use lists and to perform the basic operations on lists.

5.2.1 Basic Properties and Operations performed on Lists

Lists are the set of ordered (or ranked) records. In simulation, each record represents one entity or one event notice. Lists have –

- A top or head, indicating first item on the list
- Some way to traverse the list
- A bottom or tail, indicating last item on the list.

Every record in the list has fields like entity identifier and its attributes and also a pointer referring to next record in the list (same as singly linked list). Some lists have a pointer referring to previous record as well (same as doubly linked list). The main operations on a list include –

- Removing a recording from the top of the list (delete front operation)
- Removing a record from any location on the list
- Adding a record to the top or bottom of the list (insert front and insert rear)
- Adding a record at any arbitrary position in the list.

In the event scheduling approach, when time is advanced and the imminent event is due to be executed, the removal operation takes place – first event from the FEL is removed. Due to some priority, any in-between record may be removed. When an entity joins the queue (first in first out), addition is done at the end of the queue. If the queue has the rule *earliest due date first*, then the addition may be at any position.

For a simulation on computer, lists can be represented in two ways:

- **Arrays :** Array uses contiguous memory allocation and hence all the records are stored contiguously in the memory. So, the records can be referred using array index. But, as array uses static memory allocation, memory shortage or wastage may happen. Moreover, insertion and deletion of any record in-between needs shifting of all the records, which is time-consuming.
- **Linked Lists with dynamic memory allocation:** In most of the simulation languages, the records are dynamically created. Hence, having the list in the form of linked lists with dynamic memory allocation would be ideal. As and when a new record has to be inserted, memory can be acquired and without affecting the existing records, a new record can be inserted any where in the list.

5.3 SIMULATION IN JAVA

Java is one language which is widely used in simulation. Java does not provide any direct facilities for simulation, rather the programmer has to program all details of event-scheduling/time-advance algorithm, statistics-gathering capability, the generation of samples from probability distributions, report generator etc.

Any discrete-event simulation model written in Java consists of components like system state, entities, attributes, sets, events, activities, delays. Also, it has the following important components –

- **Clock :** A variable defining simulated time
- **Initialization method :** A method (i.e. Function) to define the system state at time 0.

- **Min-time event method :** A method that identifies the imminent event. That is, the record in the FEL, that has the smallest time stamp.
- **Event Methods:** For each event type, a method to update system state and cumulative statistics, when that event occurs.
- **Random-variate Generators:** Methods to generate samples from desired probability distributions.
- **Main program :** To maintain overall control of the event – scheduling algorithm.
- **Report generator:** A method that computes summary statistics from cumulative statistics and prints a report at the end of simulation.

The overall structure of event-scheduling simulation program in Java is shown in Figure 5.3.

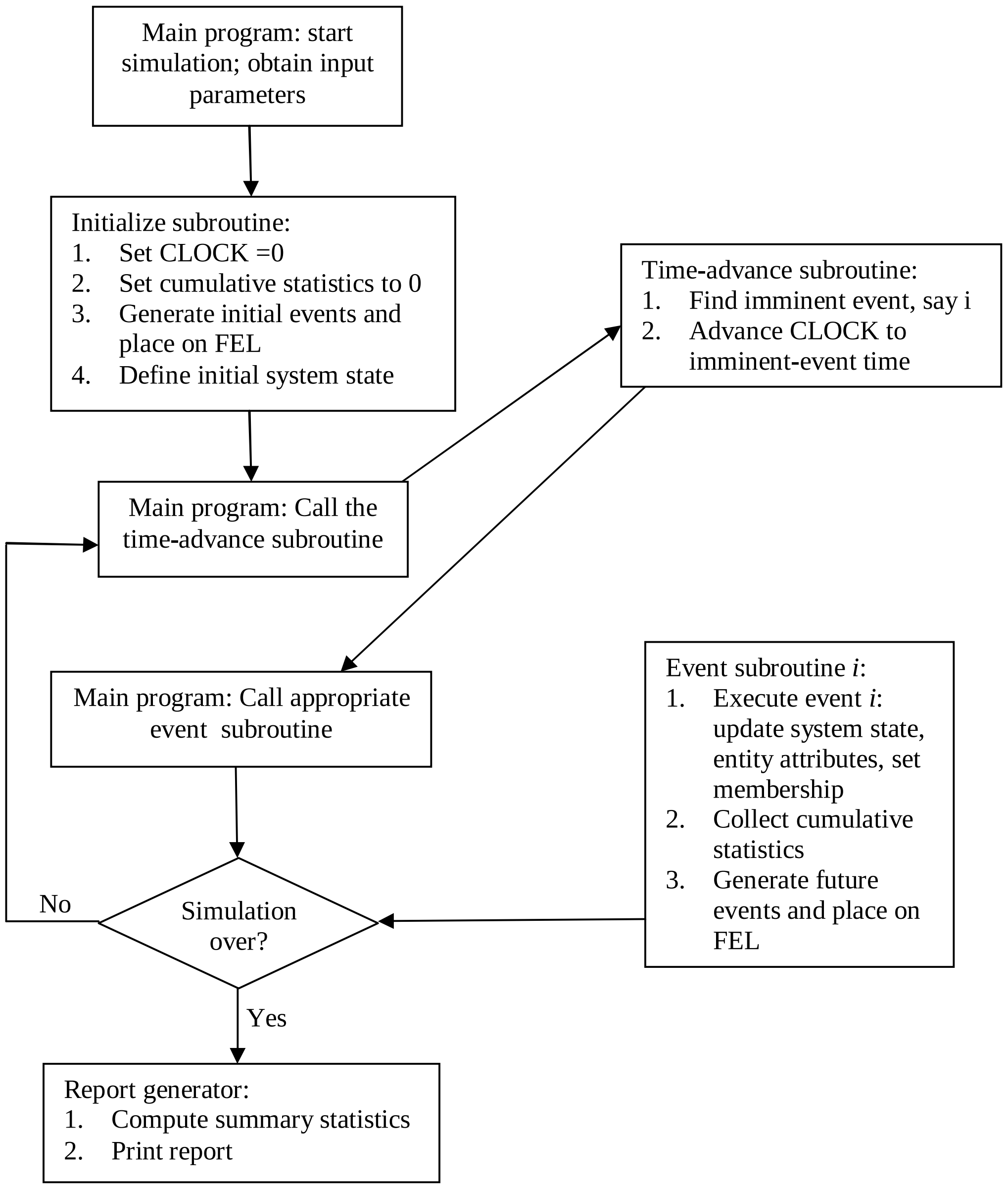
Main program: start simulation; obtain input parameters
Initialize subroutine:
1. Set CLOCK =0
2. Set cumulative statistics to 0
3. Generate initial events and place on FEL
4. Define initial system state
Time-advance subroutine:
1. Find imminent event, say i
2. Advance CLOCK to imminent-event time
Main program: Call the time-advance subroutine
Main program: Call appropriate event subroutine
Event subroutine *i*:
1. Execute event *i*: update system state, entity attributes, set membership
2. Collect cumulative statistics
3. Generate future events and place on FEL
Simulation over?
No
Yes
Report generator:
1. Compute summary statistics
2. Print report

Figure 5.3 Overall structure of event-scheduling program