System Simulation
Chapter 3: General Principles

Fatih Cavdur
fatihcavdur@uludag.edu.tr

March 29, 2014

Introduction

- This chapter develops a common framework for the modeling of complex systems by using discrete-event simulation. It covers the basic building blocks of all discrete-event simulation models: entities, attributes, activities and events.
- In discrete-event simulation, a system is modeled in terms of its state at each point in time; of the entities that pass through the system and the entities that represent system resources; and of the activities and events that cause the system state to change.

Introduction-cont.

- This chapter introduces and explains the fundamental concepts and methodologies underlying all discrete-event simulation packages.
- These are not tied to any particular package. Many of the packages use different terminology from that used in our text, and most have a number of higher-constructs designed to make modeling simpler and more straightforward for their domain.

Introduction-cont.

- For example, this chapter discusses the fundamental abstract concept of an entity, but next chapter (Chapter 4) discusses more concrete realizations of entities, such as machines, conveyors and vehicles that are built into some of the packages to facilitate modeling in the manufacturing, material handling or other domains.

Concepts in Discrete-Event Simulation

- System: A collection of entities that interact together over time to accomplish one 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.

Concepts in Discrete-Event Simulation-cont.

- Entity: Any object or component in the system that requires explicit representation in the model (e.g., a server, a customer, a machine etc.).
- Attributes: The properties of a given entity (e.g., the priority of a waiting customer, the routing of a job etc.).
- List: A collection of (permanently or temporarily) associated entities, ordered in some logical fashion (such as all customers currently waiting in a waiting line, ordered by FIFO, or by priority).

Concepts in Discrete-Event Simulation-cont.

- Event: An instantaneous occurrence that changes the state of a system (such as 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).

Concepts in Discrete-Event Simulation-cont.

- Activity (Unconditional Wait): A duration of time of specified length (e.g., a service time or inter-arrival time) which is known when it begins (although it may be defined in terms of a statistical distribution).
- Delay (Conditional Wait): A duration of time of unspecified indefinite length, which is not known until it ends (e.g., a customer's delay in a LIFO waiting line).
- Clock: A variable representing simulated time, called CLOCK in our examples.

Concepts in Discrete-Event Simulation-cont.

- Different simulation packages use different terminology for the same or similar concepts - for example, lists are sometimes called sets, queues or chains.
- Sets or lists are used to hold both entities and event notices. The entities on a list are always ordered by some rule, such as FIFO or LIFO, or some entity attribute, such as priority or due date.
- The FEL is always ranked by the event time recorded in the event notice.

Concepts in Discrete-Event Simulation-cont.

- An activity typically represents a service time, an inter-arrival time or any other processing time whose duration has been characterized and defined by the modeler. Its duration can be specified in a number of ways:
  - Deterministic
  - Statistical
  - Functional
- Note the difference between an activity and a delay.

Concepts in Discrete-Event Simulation-cont.

- The duration of an activity is computable from its specification at the instant it begins, and not affected by the occurrence of other events (unless, model contains event cancellations as allowed by some simulation packages).
- However, a delay's duration is not specified by the modeler ahead of time, but rather determined by system conditions. Quite often, we are interested in measuring it, and it is one of the desired outputs of a model run. Typically, a delay ends when some set of logical conditions becomes true, or one or more other events occur. For example, a customer's delay in a waiting line.

Example 3.1: Call Center - Revisited

System State:

$L_Q(t)$, the number of callers waiting to be served at time t;
$L_A(t)$, 0 or 1 to indicate Able as being idle or busty at t;
$L_B(t)$, 0 or 1 to indicate Baker as being idle or busty at t;

Entities:

neither callers nor servers not need to be explicitly represented, except in terms of the state variables;

Events:

arrival;
service completion by Able;
service completion by Baker;

Activities:

time between arrivals;
service time by Able;
service time by Baker;

Delays:

caller's wait in queue until Able or Baker becomes idle;

Example 3.1: Call Center - Revisited

The definition of the model components provides a static description of the model. In addition, a description of the dynamic relationships and interactions between the components is also needed. Some questions needed answers include:

- How does each event affect system state, entity attributes and set contents?
- How are activities defined (i.e., deterministic, probabilistic etc.)?
- What event marks the beginning and end of each activity?
- Can the activity begin regardless of system state?

System Simulation Chapter 3: General Principles
└Concepts in Discrete-Event Simulation

Example 3.1: Call Center - Revisited

- What events trigger the beginning (and end) of each type of delay?
- Under what conditions does a delay begin or end?
- What is the system state at time 0?
- What events should be generated at time 0 to "prime" the model - that is, to get the simulation started?

Example 3.1: Call Center - Revisited

Table : Prototype System Snapshot at Simulation Time t

Clock	System State	Entities & Attributes	Set 1	Set 2	...	FEL	Stats
t	$(x, y, z, \ldots)$	...	...	...	...	$(3, t_1)$	...
	...	...	...	...	...	$(1, t_2)$	...

Event Scheduling/Time Advance Algorithm

- The mechanism for advancing simulation time and guaranteeing that all events occur in correct chronological order is based on the FEL. This list contains all event notices for events that have been scheduled to occur at a future time.
- Scheduling a future event means that, at the instant an activity begins, its duration is computed or drawn as a sample from a distribution; and that the end-activity event, together with its event time, is placed on the FEL.

Event Scheduling/Time Advance Algorithm-cont.

- At any given time t, the FEL contains all previously scheduled future events and their associated event times.
- The FEL is ordered by event time, meaning that $t < t_1 \leq t_2 \leq \ldots \leq t_n$.
- Time t is the value of CLOCK, the current value of simulated time. After the system snapshot at simulation time CLOCK $= t$ has been updated, the CLOCK is advanced to simulation time CLOCK $= t_1$. The imminent event (the next event that will occur) notice is removed from the FEL, and the event is executed.

Event Scheduling/Time Advance Algorithm-cont.

- Execution of the imminent event means that a new system snapshot for time t_1 is created based on the old snapshot at time t and the nature of the imminent event.
- At tune t_1, new future events may or may not be generated, but if any are, they are scheduled by creating event notices and putting them into their proper position on the FEL.
- After the new system snapshot for time t_1 has been updated, the clock is advanced to the time of the new imminent event, and that event is executed.

Event Scheduling/Time Advance Algorithm-cont.

- This process repeats until the simulation is over.
- The sequence of actions that a simulator (or a simulation language) must perform to advance the clock and build a new system snapshot is called the *event-scheduling/time-advance algorithm*, whose steps are listed in the next slides.

Event Scheduling/Time Advance Algorithm-cont.

Table : Old & New System Snapshots at Time t & t_1

CLOCK	System State	...	FEL	...
t	(5 , 1 , 6)		$(3, t_1)$ $(1, t_2)$ $(1, t_3)$ ... $(2, t_n)$	
CLOCK	System State	...	FEL	...
t_1	(5 , 1 , 5)		$(1, t_2)$ $(4, t^*)$ $(1, t_3)$ ... $(2, t_n)$	

Event Scheduling/Time Advance Algorithm-cont.

Step (1) Remove the event notice for the imminent event (event 3, time t_1) from FEL.

Step (2) Advance CLOCK to imminent event time (from t to t_1).

Step (3) Execute imminent event:

- Update system state.
- Change entity attributes.
- Set membership as needed.

Step (4) Generate future events (if necessary), and place their event notices on FEL, ranked by event time. (event 4 to occur at time t^*, where, $t_2 < t^* < t_3$)

Step (5) Update cumulative statistics and counters.

System Simulation Chapter 3: General Principles
Concepts in Discrete-Event Simulation
Event Scheduling/Time Advance Algorithm

Event Scheduling/Time Advance Algorithm-cont.

- Note that the length and contents of the FEL are constantly changing during simulation. Its efficient management in a computer simulation thus has an major impact on the efficiency of the computer program representing the model.
- The management of a list is called *list processing*. The major list processing operations performed on an FEL are removal of the imminent event, addition of a new event to the list, and occasionally removal of some event (event cancellation).
- Since the imminent event is usually at the top of the list, its removal is as efficient as possible.
- Addition of a new event (and cancellation of an old event), however, requires a search of the list (and efficiency).

Bootstrapping

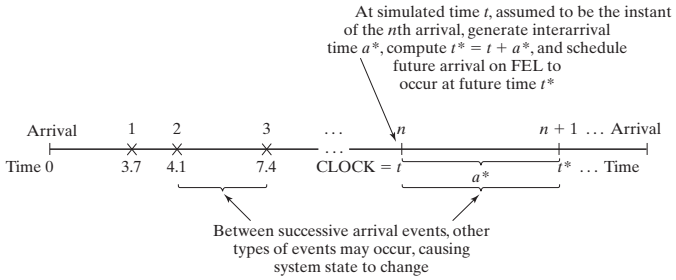

Figure : Generation of an External Arrival

World Views

- When using a simulation package or even doing a manual simulation, a modeler adopts a world view or orientation for developing a model.
- The most prevalent world views are
  - the event-scheduling world view;
  - the process-interaction world view;
  - the activity-scanning world view;

Event-Scheduling World View

- The analyst concentrate on events and their effects on system state.
- This world view will be illustrated by the manual simulations in the next slides and the Java simulation in Chapter 4.

Process-Interaction World View

- The analyst thinks in terms of processes, and defines the 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.
- More precisely, a process is the life cycle of on entity, and this life cycle consists of various events and activities.
- The process interaction is popular because it has intuitive appeal and because the simulation packages that implement it allow an analyst to describe the process flow in terms of high-level block or network constructs, while the interaction among processes is handled automatically.

Process-Interaction World View-cont.

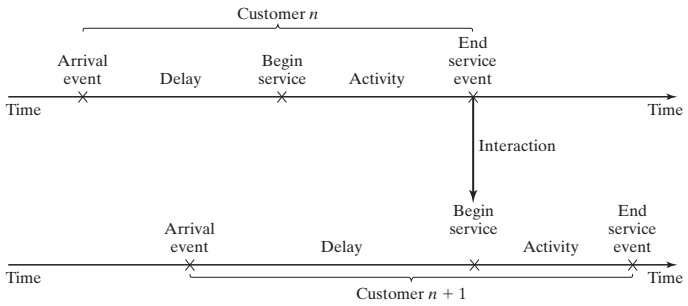

Figure : Two Interacting Customer Processes in a Single-Serve Queue

Activity-Scanning World View

- The analyst concentrates on the activities of a model and those conditions, simple or complex, that allow an activity to begin.
- At each clock advance, the conditions for each activity are checked, and, if the conditions are true, then the corresponding activity begins.
- This approach may be simple in concept and leads to modular models that are more easily maintained, understood and modified by other analysts at later times, but the repeated scanning to discover whether an activity can begin results on slow runtime on computers.

Example 3.3: Single-Channel Queue

System State:

$LQ(t)$, the number of customers waiting in queue at t;

$LS(t)$, the number of customers being served at t;

Entities:

neither customers nor server not need to be explicitly represented, except in terms of the state variables;

Events:

arrival (A);

departure (D);

stopping event (E); (scheduled to occur at time 60)

Activities:

time between arrivals;

service time;

Delays:

customer's wait in queue;

Stats:

total busy time (TBT);

maximum queue length (MQL);

Example 3.3: Single-Channel Queue

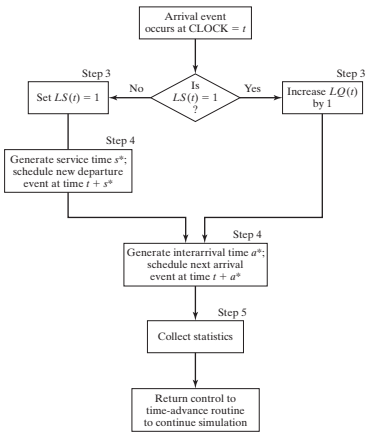

Figure : Execution of the Arrival Event

Example 3.3: Single-Channel Queue

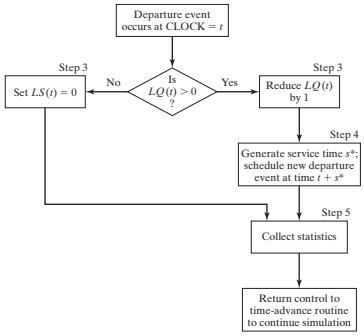

Figure : Execution of the Departure Event

Example 3.3: Single-Channel Queue

Table : Sample Simulation Table for Example 3.3

	System State				Cumulative Stats	
CLOCK	$LQ(t)$	$LS(t)$	FEL	Comment	TBT	MQL
0	0	1	(A, 1); (D, 4); (E, 60)	first A occurs (A, 0) ($a^* = 1$) schedule next A ($s^* = 4$) schedule first D	0	0

Example 3.3: Single-Channel Queue

Table : Sample Simulation Table for Example 3.3

	System State				Cumulative Stats	
CLOCK	$LQ(t)$	$LS(t)$	FEL	Comment	TBT	MQL
0	0	1	(A, 1); (D, 4); (E, 60)	first A occurs (A, 0) ($a^* = 1$) schedule next A ($s^* = 4$) schedule first D	0	0
1	1	1	(A, 2); (D, 4); (E, 60)	second A occurs (A, 1) ($a^* = 1$) schedule next A (customer delayed)	1	1

System Simulation Chapter 3: General Principles
Concepts in Discrete-Event Simulation
Manual Simulation Using Event Scheduling

Example 3.3: Single-Channel Queue

Table : Sample Simulation Table for Example 3.3

	System State				Cumulative Stats	
CLOCK	$LQ(t)$	$LS(t)$	FEL	Comment	TBT	MQL
0	0	1	(A, 1); (D, 4); (E, 60)	first A occurs (A, 0) ($a^* = 1$) schedule next A ($s^* = 4$) schedule first D	0	0
1	1	1	(A, 2); (D, 4); (E, 60)	second A occurs (A, 1) ($a^* = 1$) schedule next A (customer delayed)	1	1
2	2	1	(D, 4); (A, 8); (E, 60)	third A occurs (A, 2) ($a^* = 6$) schedule next A (two customers delayed)	2	2

Example 3.3: Single-Channel Queue

Table : Sample Simulation Table for Example 3.3

	System State				Cumulative Stats	
CLOCK	$LQ(t)$	$LS(t)$	FEL	Comment	TBT	MQL
0	0	1	(A, 1); (D, 4); (E, 60)	first A occurs (A, 0) ($a^* = 1$) schedule next A ($s^* = 4$) schedule first D	0	0
1	1	1	(A, 2); (D, 4); (E, 60)	second A occurs (A, 1) ($a^* = 1$) schedule next A (customer delayed)	1	1
2	2	1	(D, 4); (A, 8); (E, 60)	third A occurs (A, 2) ($a^* = 6$) schedule next A (two customers delayed)	2	2
4	1	1	(D, 6); (A, 8); (E, 60)	first D occurs (D, 4) ($s^* = 2$) schedule next D (customer delayed)	4	2

System Simulation Chapter 3: General Principles
Concepts in Discrete-Event Simulation
Manual Simulation Using Event Scheduling

Example 3.3: Single-Channel Queue

Table : Sample Simulation Table for Example 3.3

	System State				Cumulative Stats	
CLOCK	$LQ(t)$	$LS(t)$	FEL	Comment	TBT	MQL
0	0	1	(A, 1); (D, 4); (E, 60)	first A occurs (A, 0) ($a^* = 1$) schedule next A ($s^* = 4$) schedule first D	0	0
1	1	1	(A, 2); (D, 4); (E, 60)	second A occurs (A, 1) ($a^* = 1$) schedule next A (customer delayed)	1	1
2	2	1	(D, 4); (A, 8); (E, 60)	third A occurs (A, 2) ($a^* = 6$) schedule next A (two customers delayed)	2	2
4	1	1	(D, 6); (A, 8); (E, 60)	first D occurs (D, 4) ($s^* = 2$) schedule next D (customer delayed)	4	2
6	0	1	(A, 8); (D, 11); (E, 60)	second D occurs (D, 6) ($s^* = 5$) schedule next D	6	2

Example 3.3: Single-Channel Queue

Table : Sample Simulation Table for Example 3.3

	System State				Cumulative Stats	
CLOCK	$LQ(t)$	$LS(t)$	FEL	Comment	TBT	MQL
0	0	1	(A, 1); (D, 4); (E, 60)	first A occurs (A, 0) ($a^* = 1$) schedule next A ($s^* = 4$) schedule first D	0	0
1	1	1	(A, 2); (D, 4); (E, 60)	second A occurs (A, 1) ($a^* = 1$) schedule next A (customer delayed)	1	1
2	2	1	(D, 4); (A, 8); (E, 60)	third A occurs (A, 2) ($a^* = 6$) schedule next A (two customers delayed)	2	2
4	1	1	(D, 6); (A, 8); (E, 60)	first D occurs (D, 4) ($s^* = 2$) schedule next D (customer delayed)	4	2
6	0	1	(A, 8); (D, 11); (E, 60)	second D occurs (D, 6) ($s^* = 5$) schedule next D	6	2
8	1	1	(D, 11); (A, 11); (E, 60)	fourth A occurs (A, 8) ($a^* = 3$) schedule next A (customer delayed)	8	2

Example 3.4: Dump Truck Problem

System State:

$LQ(t)$, number of trucks in loader queue at t;

$L(t)$, number of trucks (0, 1 or 2) being loaded at t;

$WQ(t)$, number of trucks in weigh queue at t;

$W(t)$, number of trucks (0 or 1) being weighed at t;

Entities:

$DT1, \ldots DT6$;

Events:

(ALQ, t, DT_i): dump truck i arrives at loader queue (ALQ) at time t;

(EL, t, DT_i): dump truck i ends loading (EL) at time t;

(EW, t, DT_i): dump truck i ends weighing (EW) at time t;

Lists:

loader queue, weigh queue;

Activities:

loading time, weighing time, travel time;

Delays:

wait in loader queue, delay at scale;

Stats:

total busy time of (both) loaders (B_L) and scale (B_S) from time 0 to time t;

Example 3.4: Dump Truck Problem

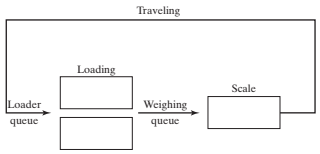

Figure : Dump Truck Problem

Example 3.4: Dump Truck Problem

Table : Sample Simulation Table for Example 3.4

t	$LQ(t)$	$L(t)$	$WQ(t)$	$W(t)$	List 1 (Q1)	List 2 (Q2)	FEL	B_L	B_S
0	3	2	0	1	DT4		(EL, 5 , DT3)	0	0
					DT5		(EL, 10 , DT2)		
					DT6		(EW, 12 , DT1)		

Example 3.4: Dump Truck Problem

Table : Sample Simulation Table for Example 3.4

t	$LQ(t)$	$L(t)$	$WQ(t)$	$W(t)$	List 1 (Q1)	List 2 (Q2)	FEL	B_L	B_S
0	3	2	0	1	DT4		(EL, 5 , DT3)	0	0
					DT5		(EL, 10 , DT2)		
					DT6		(EW, 12 , DT1)		
5	2	2	1	1	DT5	DT3	(EL, 10 , DT2)	10	5
					DT6		(EL, 5 + 5 , DT4)		
							(EW, 12 , DT1)		

System Simulation Chapter 3: General Principles
Concepts in Discrete-Event Simulation
Manual Simulation Using Event Scheduling

Example 3.4: Dump Truck Problem

Table : Sample Simulation Table for Example 3.4

t	$LQ(t)$	$L(t)$	$WQ(t)$	$W(t)$	List 1 (Q1)	List 2 (Q2)	FEL	B_L	B_S
0	3	2	0	1	DT4		(EL, 5 , DT3)	0	0
					DT5		(EL, 10 , DT2)		
					DT6		(EW, 12 , DT1)		
5	2	2	1	1	DT5	DT3	(EL, 10 , DT2)	10	5
					DT6		(EL, 5 + 5 , DT4)		
							(EW, 12 , DT1)		
10	1	2	2	1	DT6	DT3	(EL, 10 , DT4)	20	10
						DT2	(EW, 12 , DT1)		
							(EL, 10 + 10 , DT5)		

Example 3.4: Dump Truck Problem

Table : Sample Simulation Table for Example 3.4

t	$LQ(t)$	$L(t)$	$WQ(t)$	$W(t)$	List 1 (Q1)	List 2 (Q2)	FEL	B_L	B_S
0	3	2	0	1	DT4		(EL, 5 , DT3)	0	0
					DT5		(EL, 10 , DT2)		
					DT6		(EW, 12 , DT1)		
5	2	2	1	1	DT5	DT3	(EL, 10 , DT2)	10	5
					DT6		(EL, 5 + 5 , DT4)		
							(EW, 12 , DT1)		
10	1	2	2	1	DT6	DT3	(EL, 10 , DT4)	20	10
						DT2	(EW, 12 , DT1)		
							(EL, 10 + 10 , DT5)		
10	0	2	3	1		DT3	(EW, 12 , DT1)	20	10
						DT2	(EL, 20 , DT5)		
						DT4	(EL, 10 + 15 , DT6)		

System Simulation Chapter 3: General Principles
Concepts in Discrete-Event Simulation
Manual Simulation Using Event Scheduling

Example 3.4: Dump Truck Problem

Table : Sample Simulation Table for Example 3.4

t	$LQ(t)$	$L(t)$	$WQ(t)$	$W(t)$	List 1 (Q1)	List 2 (Q2)	FEL	B_L	B_S
0	3	2	0	1	DT4		(EL, 5 , DT3)	0	0
					DT5		(EL, 10 , DT2)		
					DT6		(EW, 12 , DT1)		
5	2	2	1	1	DT5	DT3	(EL, 10 , DT2)	10	5
					DT6		(EL, 5 + 5 , DT4)		
							(EW, 12 , DT1)		
10	1	2	2	1	DT6	DT3	(EL, 10 , DT4)	20	10
						DT2	(EW, 12 , DT1)		
							(EL, 10 + 10 , DT5)		
10	0	2	3	1		DT3	(EW, 12 , DT1)	20	10
						DT2	(EL, 20 , DT5)		
						DT4	(EL, 10 + 15 , DT6)		
12	0	2	2	1		DT2	(EL, 20 , DT5)	24	12
						DT4	(EW, 12 + 12 , DT3)		
							(EL, 25 , DT6)		
							(ALQ, 12 + 60 , DT1)		

Lists: Basic Properties and Operations

- Lists are set of ordered or ranked records. In simulation, each record represents one entity or one event notice.
- Lists are ranked, so they have a *top* or *head* (the first item on the list), and a *bottom* or *tail* (the last item on the list).
- A *head pointer* is a variable that points to or indicates the record at the top of the list. Some implementations also have a *tail pointer*.
- An entity, along with its attributes or an event notice, will be referred to as a *record*. An entity identifier and its attributes are *fields* in the entity record; the event type, event time and any other event-related data are fields in the event-notice record.

Lists: Basic Properties and Operations-cont.

The main operations on a list are the following:

- Removing a record from the top of the list.
- Removing a record from any location on the list.
- Adding an entity record to the top or bottom of the list.
- Adding a record at an arbitrary position in the list specified by the ranking rule.

Using Arrays for List Processing

- The array method of list storage is typical of FORTRAN, but it may be used in other procedural languages.
- For convenience, we use the notation $R(i)$ to refer to the i-th record in the array, however, it may be appropriately stored in the language being used.
- Arrays are advantageous in that any specified record, say i-th, can be retrieved without searching.
- They are disadvantageous when items are added in the middle of a list or the list must be rearranged.

Using Dynamic Allocation and Linked Lists

- In procedural languages, such as C++ and Java, and in most simulation languages, entity records are dynamically created when an entity is created and event notice records are dynamically created whenever an event is scheduled on the future event list.
- With dynamic allocation, a record is referenced by a pointer instead of by array index.
- In C++ or Java, pointers can be used to access to the allocated record.

Advanced Techniques

- Many of the modern simulation packages use more efficient approaches than searching through a doubly-linked list.
- We will not focus on the details of these approaches, but some of them are
  - Using a middle pointer in addition to the head and tail pointers.
  - Some advanced algorithms use list representations other than a doubly-linked list, such as heaps or trees.

Summary

- Reading HW: Chapter 3.
- Chapter 3 Exercises.