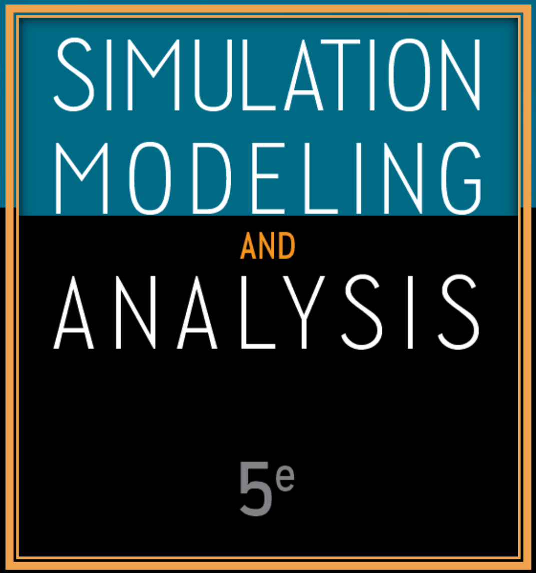
Slides Created by Jennifer Peterson
Chimborazo Publishing, Inc.




# Chapter 1

# Basic Simulation Modeling

Averill M. Law

# 1.1 The Nature of Simulation

- Simulation
  - Using computer techniques to simulate a real-world facility or process (the system)
- Model
  - Mathematical or logical representation of system behavior
- Model is evaluated numerically in a simulation

# 1.2 Systems, Models, and Simulation

- Types of systems
  - Discrete
    - State variables change at separated points in time
  - Continuous
    - State variables change continuously with time

# Systems, Models, and Simulation



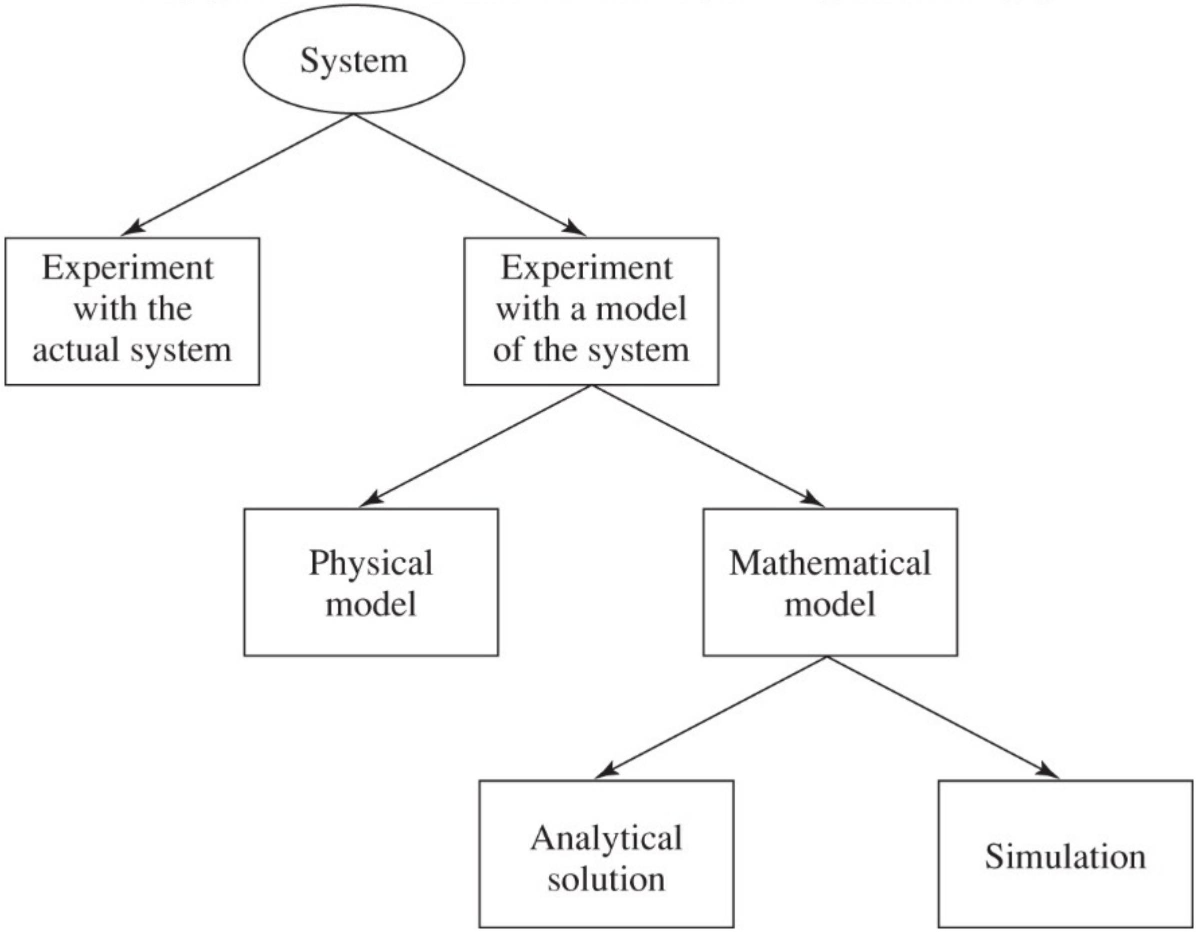



Figure 1.1 Ways to study a system

# 1.3 Discrete-Event Simulation

- Event
  - An instantaneous occurrence that may change the system's state
- Simulation clock
  - Variable that gives the current value of simulated time
  - Methods for advancing the simulation clock
    - Next-event time advance
    - Fixed-increment time advance

# Discrete-Event Simulation

- Next-event time advance
  - Clock initialized to zero
  - Times of events are determined
  - Clock advanced to time of first event
    - State of system updated
  - Clock advanced to time of next event
    - State of system updated
  - Continues until stopping condition satisfied

# Discrete-Event Simulation

- Fixed-increment time advance
  - Does not skip over inactive periods
    - Can use up a lot of computer time

# 1.4 Simulation of a Single-Server Queuing System

- Example system: a one-operator barbershop
  - Interarrival times $A_1, A_2 \ldots A_n$ are independent, identically distributed random variables
  - Customer service times are $S_1, S_2 \ldots$
  - Estimate three quantities
    - Expected average delay of customers in queue
    - Expected average number of customers in queue
    - Expected utilization of the server

# Simulation of a Single-Server Queuing System



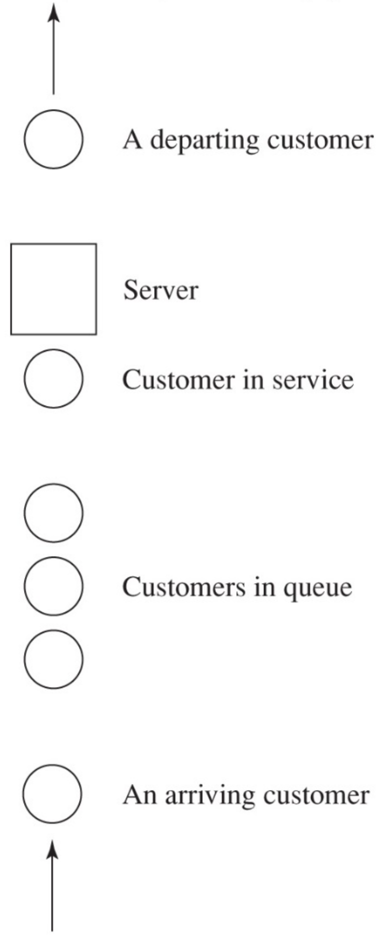


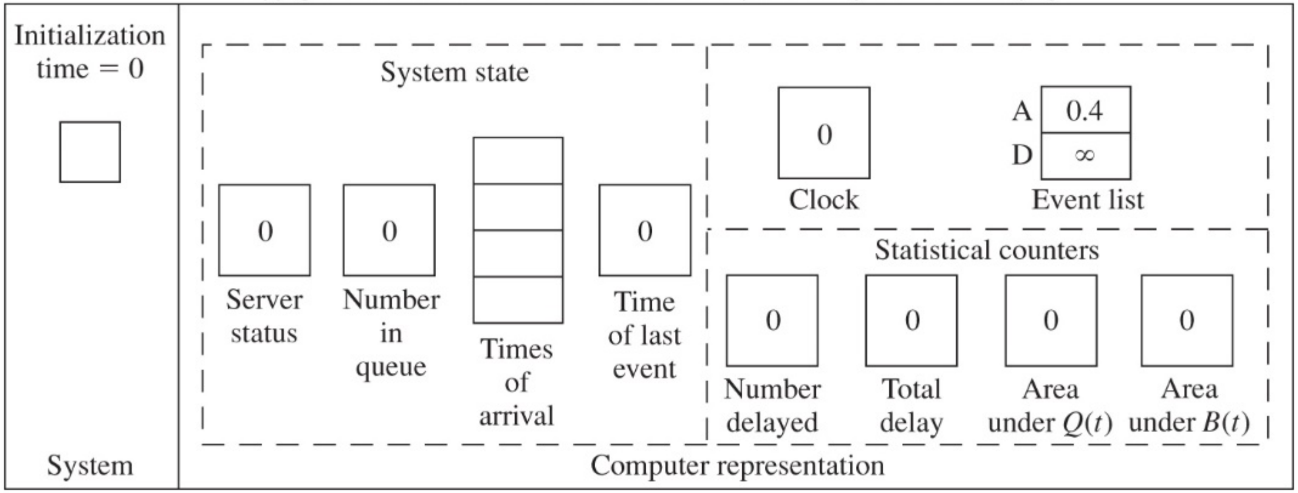

Figure 1.4 A single-server queuing system

# Simulation of a Single-Server Queuing System

- Events for the barbershop example
  - Arrival time of a customer
  - Departure time of a customer
- State variables
  - Status of the server
  - Number of customers in the queue
  - Arrival time of each customer currently in queue
  - Time of most recent event

# Simulation of a Single-Server Queuing System

- Initialization
  - State of the system at $t = 0$
- Sequence of events
  - $t = 0.4$ arrival of customer 1
  - $t = 1.6$ arrival of customer 2
  - $t = 2.1$ arrival of customer 3
  - $t = 2.4$ departure of customer 1





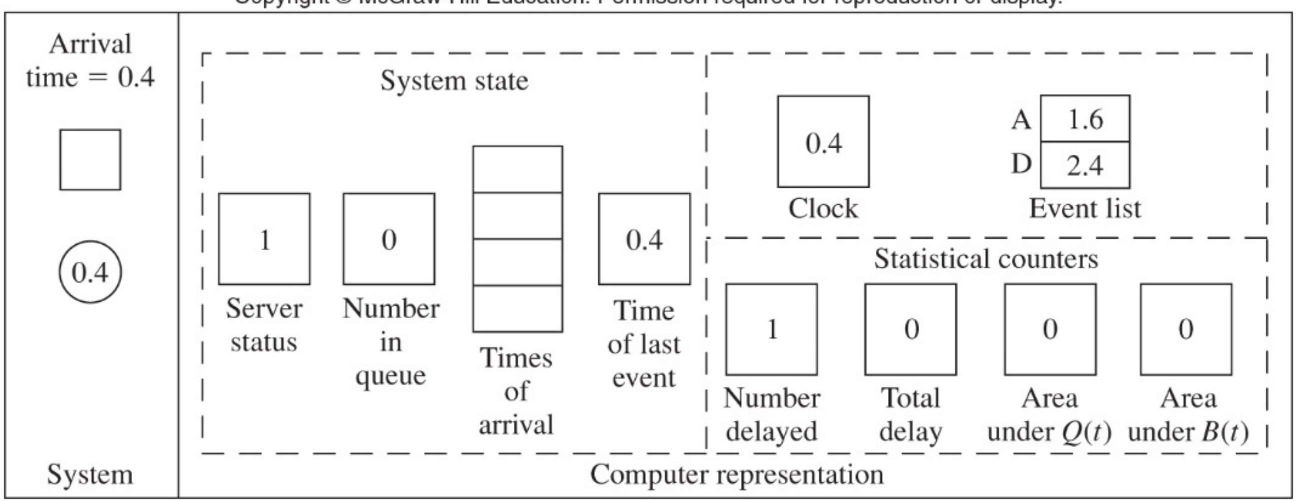
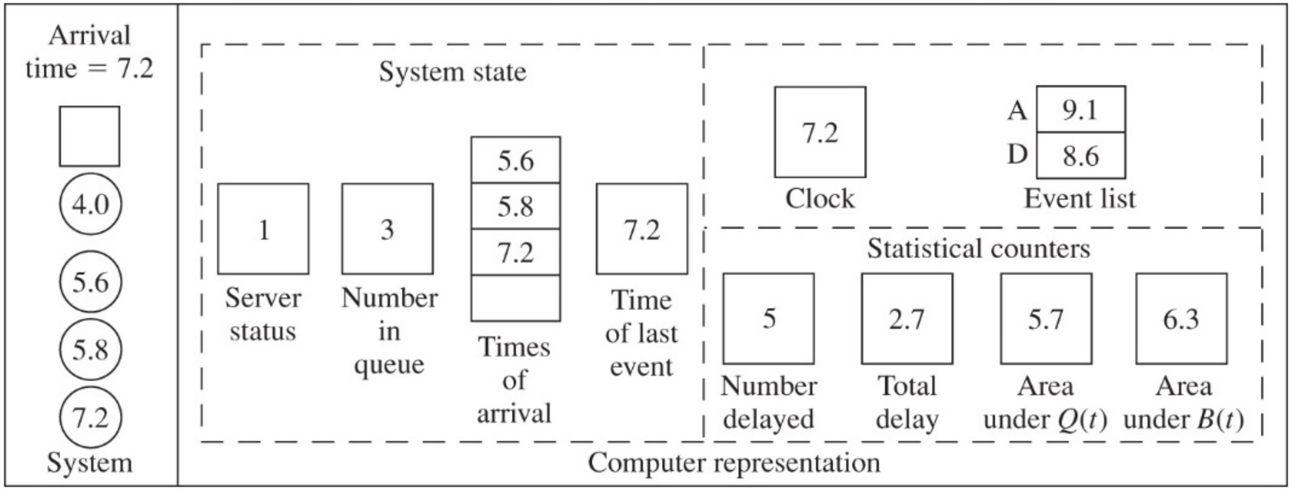




Figure 1.7 (a-b) Snapshots of the system and its computer representation at times t=0 and t=0.4





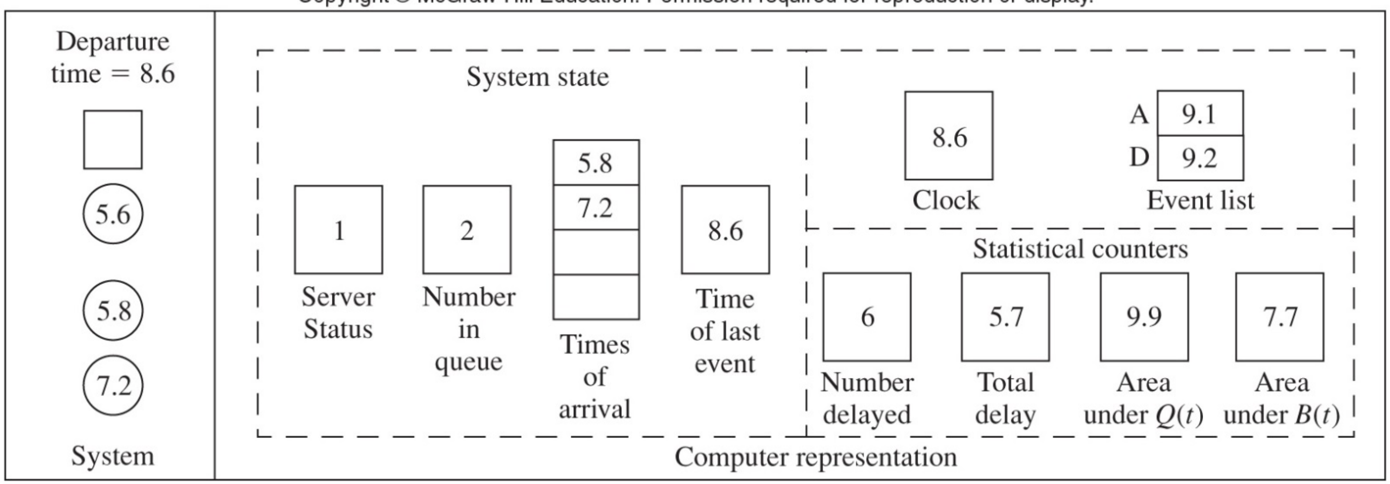




Figure 1.7 (m-n) Snapshots of the system and its computer representation at times t=7.2 and t=8.6

# 1.4 Simulation of a Single-Server Queuing System

- Writing a simulation program
  - Our example: C, a general purpose language
  - Program modules
    - Initialization
    - Timing
    - Arrive
    - Depart
    - Report

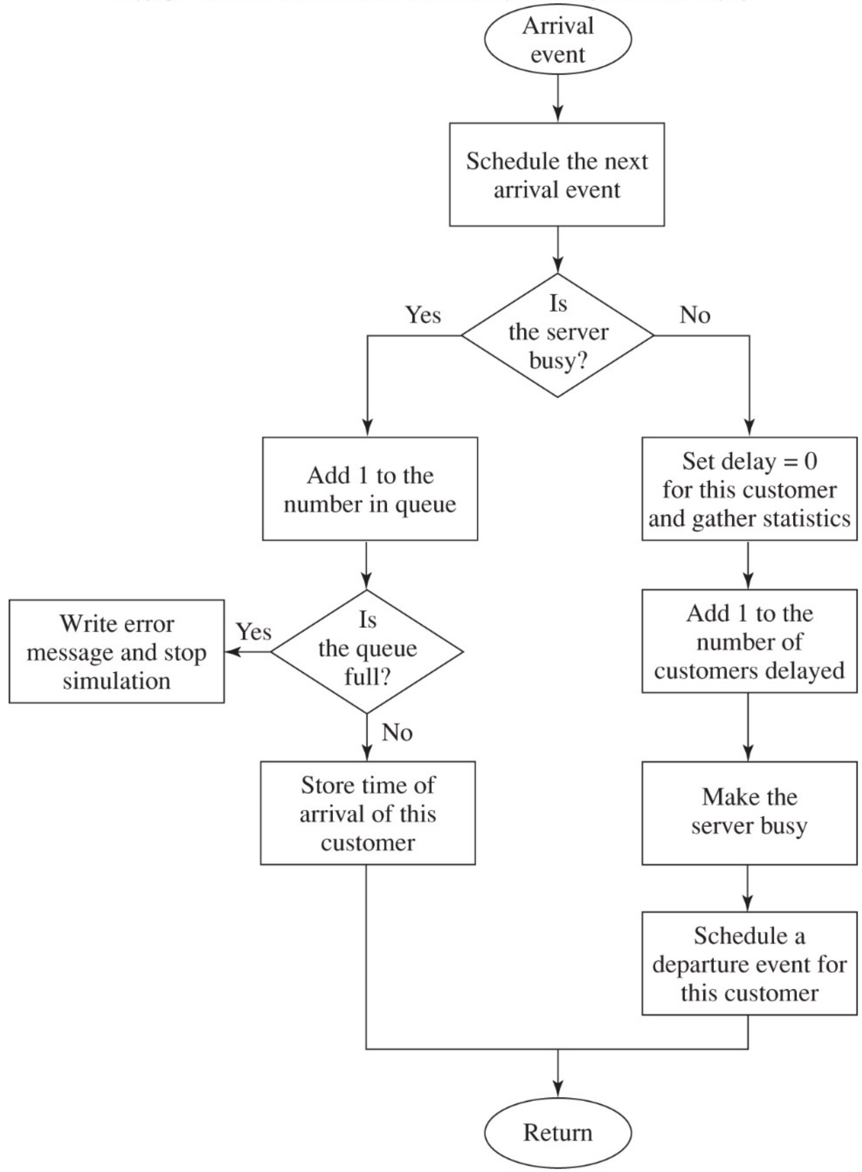



Figure 1.8 Flowchart for arrival routine – queuing model

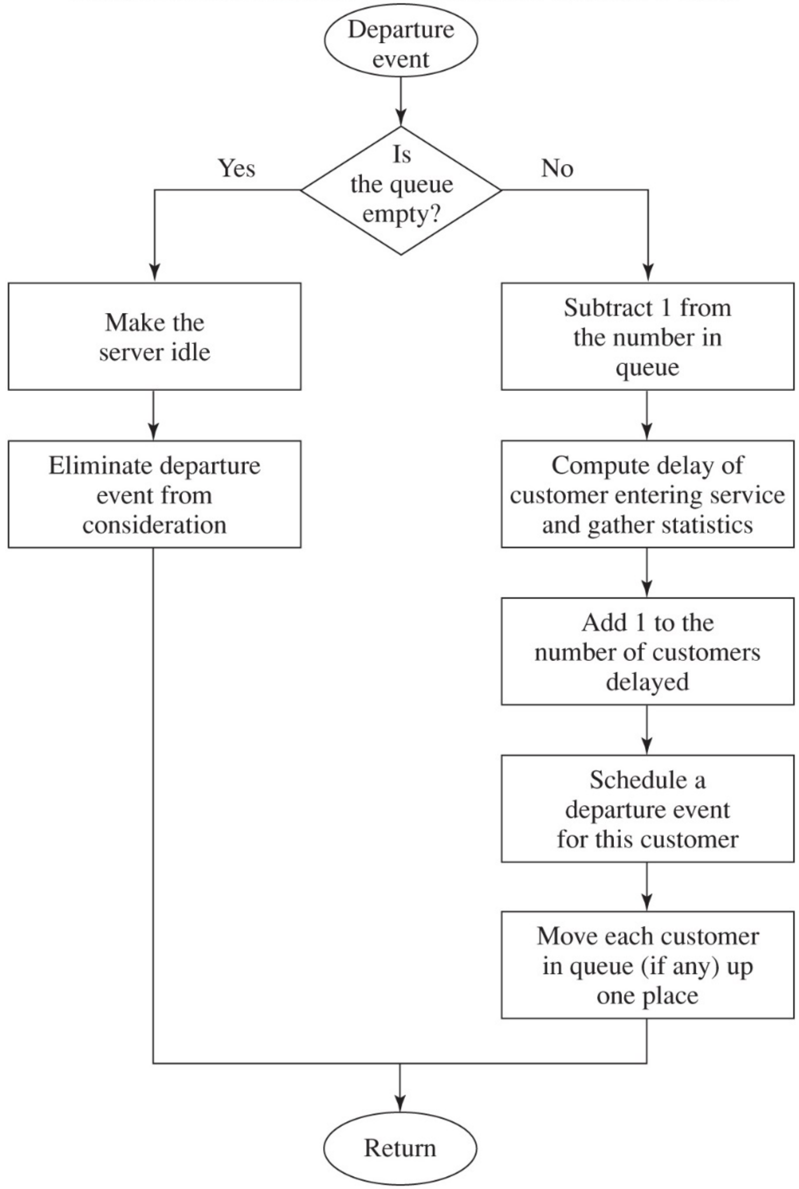



Figure 1.9 Flowchart for departure routine – queuing model

# Simulation of a Single-Server Queuing System



```
/* External definitions for single-server queueing system. */

#include <stdio.h>
#include <math.h>
#include "lcgrand.h"  /* Header file for random-number generator. */

#define Q_LIMIT 100  /* Limit on queue length. */
#define BUSY      1  /* Mnemonics for server's being busy */
#define IDLE      0  /* and idle. */

int   next_event_type, num_custs_delayed, num_delays_required, num_events,
      num_in_q, server_status;
float area_num_in_q, area_server_status, mean_interarrival, mean_service,
      sim_time, time_arrival[Q_LIMIT + 1], time_last_event, time_next_event[3],
      total_of_delays;
FILE  *infile, *outfile;

void  initialize(void);
void  timing(void);
void  arrive(void);
void  depart(void);
void  report(void);
void  update_time_avg_stats(void);
float expon(float mean);
```

Figure 1.10 C code for the external definitions, queuing model

```
main()  /* Main function. */
{
    /* Open input and output files. */

    infile  = fopen("mm1.in",  "r");
    outfile = fopen("mm1.out", "w");

    /* Specify the number of events for the timing function. */

    num_events = 2;

    /* Read input parameters. */

    fscanf(infile, "%f %f %d", &mean_interarrival, &mean_service,
           &num_delays_required);

    /* Write report heading and input parameters. */

    fprintf(outfile, "Single-server queueing system\n\n");
    fprintf(outfile, "Mean interarrival time%11.3f minutes\n\n",
            mean_interarrival);
    fprintf(outfile, "Mean service time%16.3f minutes\n\n", mean_service);
    fprintf(outfile, "Number of customers%14d\n\n", num_delays_required);

    /* Initialize the simulation. */

    initialize();
```

Figure 1.11 C code for the main function, queuing model (continues)

```c
    /* Run the simulation while more delays are still needed. */

    while (num_custs_delayed < num_delays_required) {

        /* Determine the next event. */

        timing();

        /* Update time-average statistical accumulators. */

        update_time_avg_stats();

        /* Invoke the appropriate event function. */

        switch (next_event_type) {
            case 1:
                arrive();
                break;
            case 2:
                depart();
                break;
        }
    }

    /* Invoke the report generator and end the simulation. */

    report();

    fclose(infile);
    fclose(outfile);

    return 0;
}
```

Figure 1.11 C code for the main function, queuing model (cont'd.)

# Simulation of a Single-Server Queuing System



```
void initialize(void)  /* Initialization function. */
{
    /* Initialize the simulation clock. */

    sim_time = 0.0;

    /* Initialize the state variables. */

    server_status   = IDLE;
    num_in_q        = 0;
    time_last_event = 0.0;

    /* Initialize the statistical counters. */

    num_custs_delayed  = 0;
    total_of_delays    = 0.0;
    area_num_in_q      = 0.0;
    area_server_status = 0.0;

    /* Initialize event list.  Since no customers are present, the departure
       (service completion) event is eliminated from consideration. */

    time_next_event[1] = sim_time + expon(mean_interarrival);
    time_next_event[2] = 1.0e+30;
}
```

Figure 1.12 C code for function initialize, queuing model

# Simulation of a Single-Server Queuing System



```
void timing(void)  /* Timing function. */
{
    int   i;
    float min_time_next_event = 1.0e+29;

    next_event_type = 0;

    /* Determine the event type of the next event to occur. */

    for (i = 1; i <= num_events; ++i)
        if (time_next_event[i] < min_time_next_event) {
            min_time_next_event = time_next_event[i];
            next_event_type     = i;
        }

    /* Check to see whether the event list is empty. */

    if (next_event_type == 0) {

        /* The event list is empty, so stop the simulation. */

        fprintf(outfile, "\nEvent list empty at time %f", sim_time);
        exit(1);
    }

    /* The event list is not empty, so advance the simulation clock. */

    sim_time = min_time_next_event;
}
```

Figure 1.13 C code for function timing, queuing model

# Simulation of a Single-Server Queuing System

- Discussion of simulation output
  - Numbers will vary each time the simulation is run
    - Not explicit answers but estimates of quantities
  - Results are functions of the input parameters, and they way system is initialized
  - Might want to study steady state characteristics of the system
  - Alternative stopping rules could have been defined

# Simulation of a Single-Server Queuing System



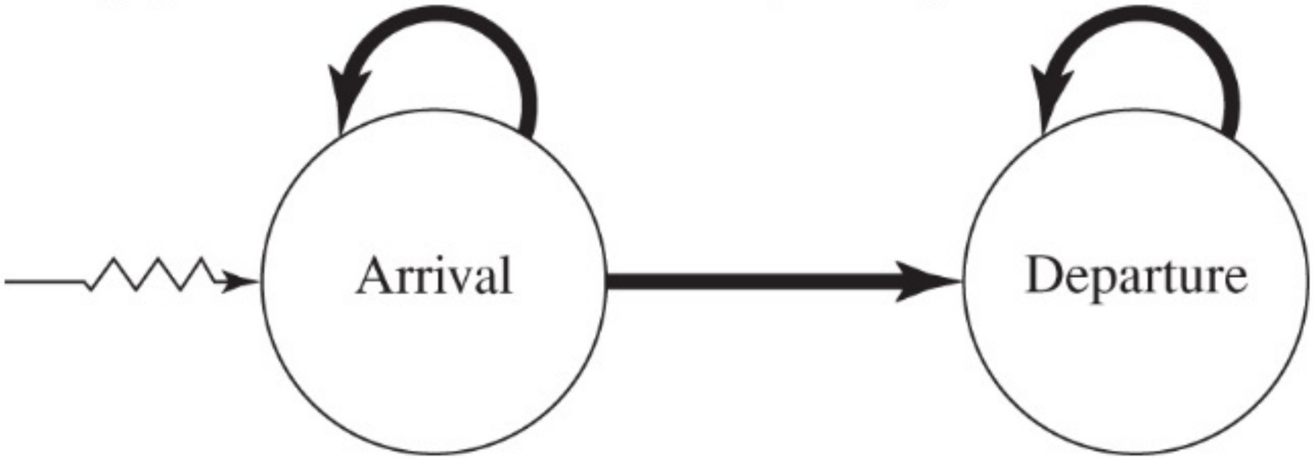



Figure 1.25 Event graph, queuing model

# 1.5 Simulation of an Inventory System

- Problem: compare various ordering policies for an inventory system
  - Given: initial inventory level, demands, times between demands
  - Costs: setup cost, incremental cost, holding and shortage costs
  - State variables: inventory level, amount of an outstanding order from company to supplier, and time of last event

# Simulation of an Inventory System

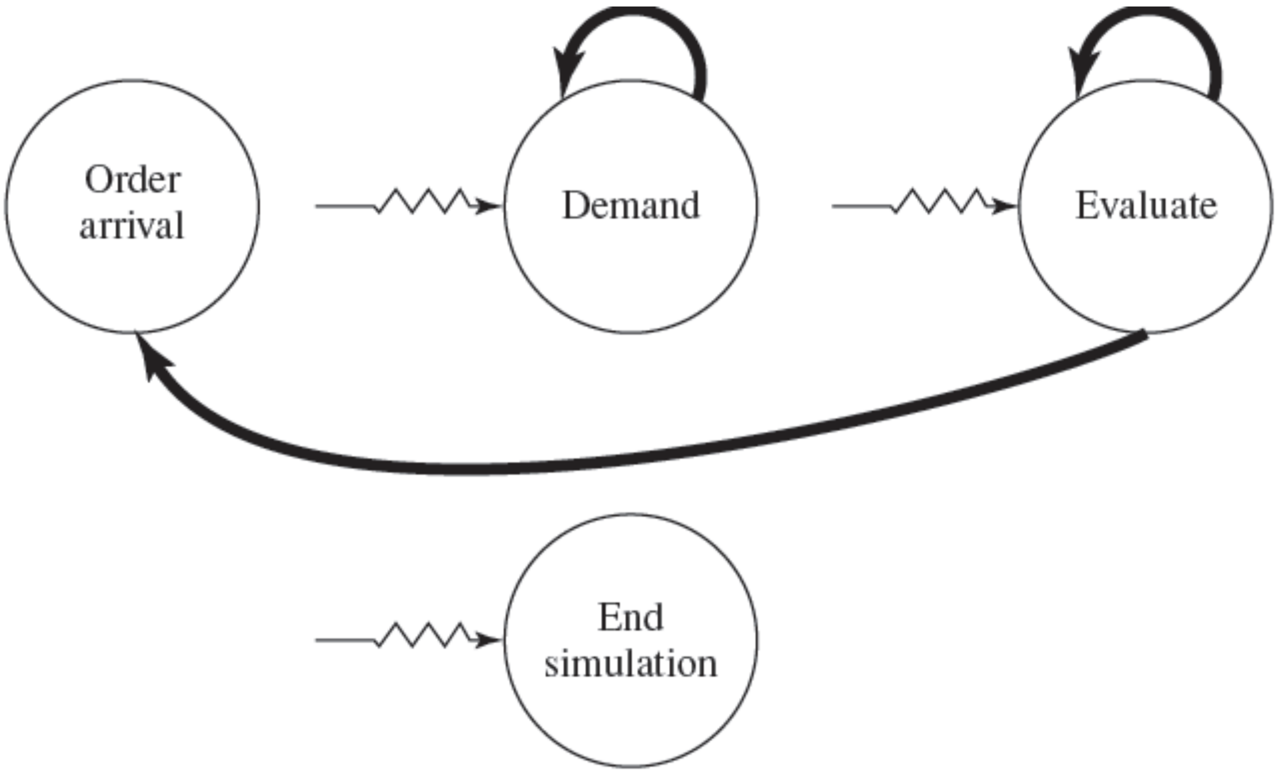



Figure 1.29 Event graph, inventory model

# 1.6 Parallel/Distributed Simulation and the High Level Architecture

- Parallel-discrete event simulation
  - Execution of the simulation using multiple processors
  - Reduces execution time
  - Done by dividing model into several logical processes (LPs)
  - Critical issue: determining LPs happen in proper sequence

# Parallel/Distributed Simulation and the High Level Architecture

- Types of synchronization in parallel simulation
  - Conservative
    - Avoid any violations of local causality constraint
  - Optimistic
    - Time-warp mechanism: best known optimistic approach
- Distributed simulation
  - HLA federation

# Parallel/Distributed Simulation and the High Level Architecture

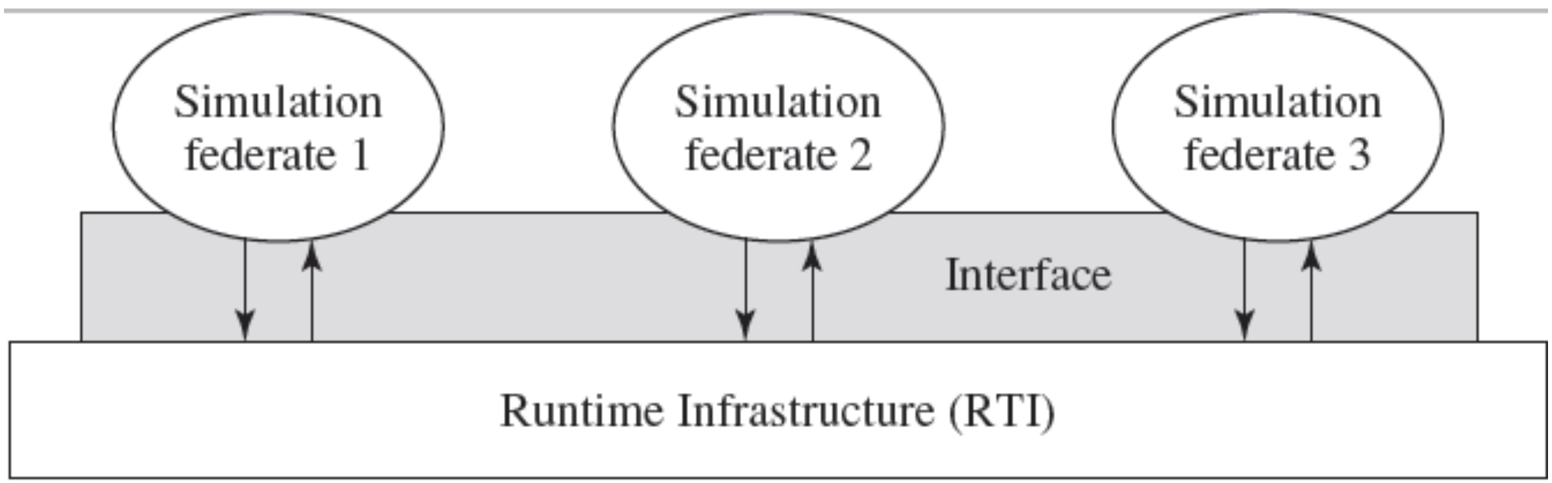



Figure 1.45 Functional view of an HLA federation

# 1.7 Steps in a Sound Simulation Study

- Formulate the problem and plan the study
- Collect data and define a model
- Ensure the assumptions are valid
- Construct a computer program and verify
- Make the pilot runs
- Is the programmed model valid?
- Design the experiments

# Steps in a Sound Simulation Study

- Make the production runs
- Analyze the output data
- Document, present, and use the results

# 1.8 Advantages, Disadvantages, and Pitfalls of Simulation

- Most complex, real-world systems cannot be accurately described by an analytical mathematical model
  - Numerical simulation is the only investigation possible
- Simulation allow for the:
  - Evaluation of alternative designs
  - Study of a system with a long time frame

# Advantages, Disadvantages, and Pitfalls of Simulation

- Simulation models can be expensive and time-consuming to develop
- Large amounts of data can lead to "overconfidence" in the result
- What are some causes of failure?
  - Lack of well-defined objectives
  - Inappropriate level of detail in the model
- Crucial to involve the right people in the simulation study