

# ***Simulation***



***By: Professor M.R.Asharif***

---

***Department of Information Eng.***

***University of the Ryukyus***



# Basic Simulation Modeling

## ***Definition of a system:***

---

“The facility or process of interest called a system”

## ***Definition of a model:***

“Assumption which usually take the form of mathematical or logical relationship, constitute a model”

## ***Definition of a simulation:***

“To use computer to imitate and evaluate a model numerically”



# Simulation Drawbacks

- 1-Complexity of writing computer programs (needs development of excellent software).
- 2-Large amount of computer time (becoming less severe as computing cost decreases).
- 3-Not considering of all aspects of real model (methodology is more important than just software writing and running) .



# Application areas for Simulation

---

- Designing and analyzing manufacturing systems
- Evaluation hard and software for computer
- Evaluation a new military weapons system or tactic
- Ordering policies for an inventory system
- Designing communications and message protocols
- Designing transportations facilities, airport, subways
- Designing organizations: hospitals, post offices, restaurants
- Analyzing financial and economic systems, stock share, Exchange rate, etc.
- As a technique, simulation is powerful tool in operations research and management science.

# Systems, Models, and Simulation

## System definition by Schmidt and Taylor(1970):

- A collection of entities, people, machines that act and interact together to accomplish a logical end.
- Determining the number of tellers in a Banking system is different if one considers for only cashing, deposit, loan, etc. or all together.
- State of a system is defined as: The collection of variables to describe the system.
- In Bank state variables are: 1-Number of busy tellers 2- Number of customers 3- Time

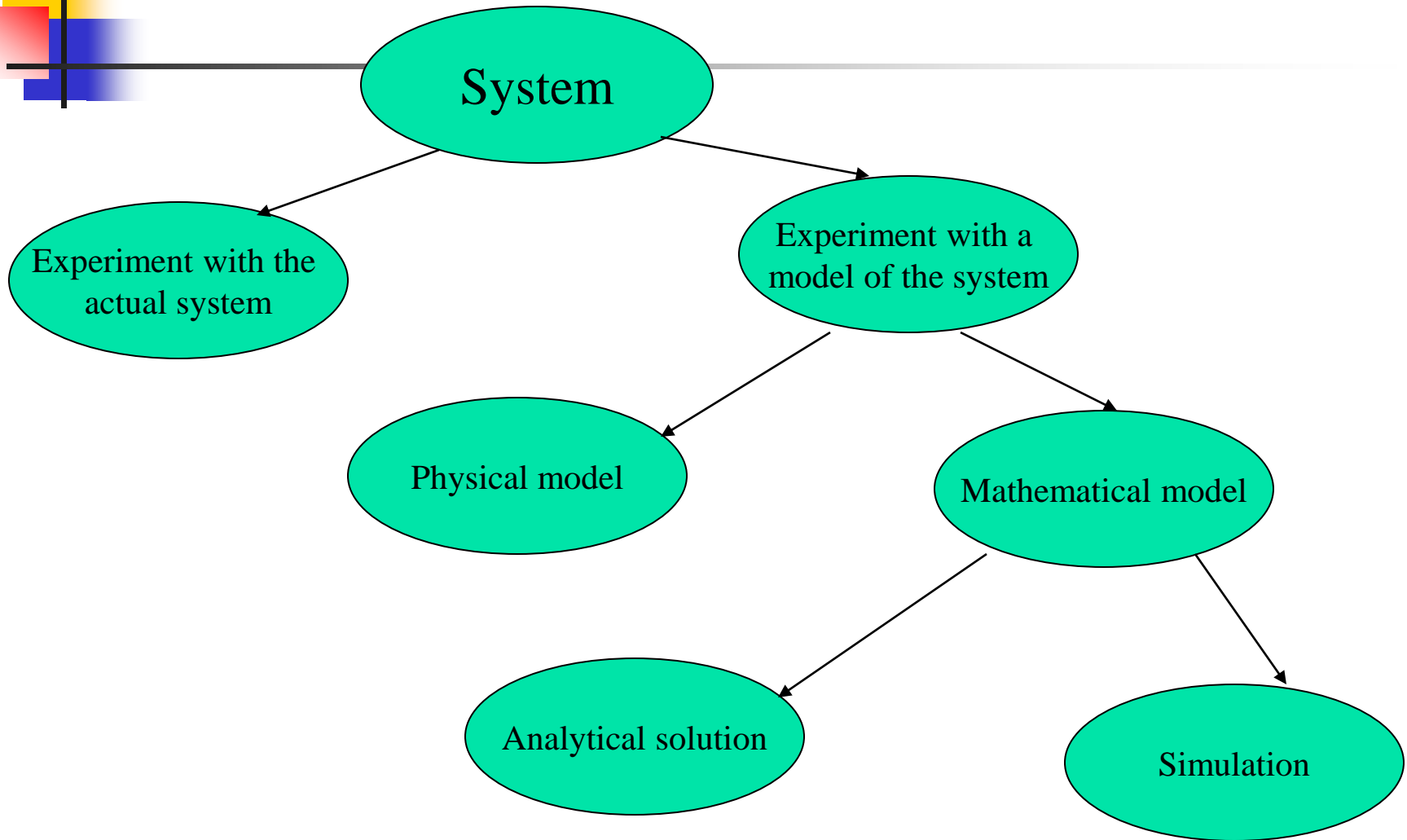


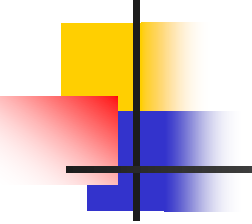
# Types of System

---

- 1- Discrete System: The state variables change at separated points in time: Number of customers in Bank.
- 2- Continuous System: The state variables change continuously with time: Position and velocity of an airplane at any time.

# System Study



- 
- Experiment with the actual system vs. Experiment with a model of the system
  - Actual System: It is too costly or disruptive to change system physically
  - Model of the System: However, system's model validity always should be checked





## Physical Model vs. mathematical Model

---

- A cockpit disconnected from airplane for pilot training is a physical model (iconic model). A physical model is useful to study engineering system.
- But mathematical model can also represent the system in terms of logical and relations, if it is valid one.  $x=v.t$  is good model but can not be used in rush-hour commuting on congested urban highways.



## Analytical Solution vs. Simulation

---

- After making mathematical model it might be easy to solve it to get the answer. But, if the analytical solution is complex (inverting nonsparse matrix) it needs vast computing resources.
- Then, solving by simulation might give us a more rapid answer. That is finding numerical input-output relation.



## Simulation Model

---

- Static vs. Dynamic Simulation Models: model of the system.
- Static simulation model: Time does not play any role: Monte Carlo Models.
- Dynamic simulation model: A system that evolves over time.



# Simulation Model

---

- Deterministic vs. Stochastic Simulation Models:
- Deterministic model: does not contain random components. The output is determined once input is cleared.
- Stochastic model: contains random input. Then output is random too. So, it is an estimation of system.



# Simulation Model

---

- Continuous vs. Discrete Simulation Models:
- Continuous model for modeling continuous system.
- Discrete model for modeling discrete system.



# Discrete-Event Simulation Models

---

- The simulation of the system that evolve over time (Dynamic), in which the state variables change at separate point in time (Discrete) with some stochastic (random) variables.



# Single Serve Example

- Consider a single server facility such as one-operator barbershop information desk at an airport.
- We want to estimate the average delay in queue (waiting time for service).
- The state variables are: 1- Status of server (idle or busy) 2- Customers number waiting in queue. 3- Time of arrival



# Single Serve Example

---

- Two types of events for this system are:
  - 1- Event of arrival of customer. Status of server changes from idle to busy or queue increments by one.
  - 2- Event of completion of service for a customer resulting in the customer's departure. Server's status changes from busy to idle or queue decrements by one.





# Time-Advance Mechanisms

---


- The current value of simulation time is called the simulation clock. The simulation clock advances: 1- next-event time advance. 2- fixed-increment time advance.
- In (1), the simulation clock jump to the first event in future, the state of the system is updated.
- (2) is special case of (1).

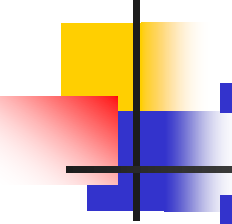


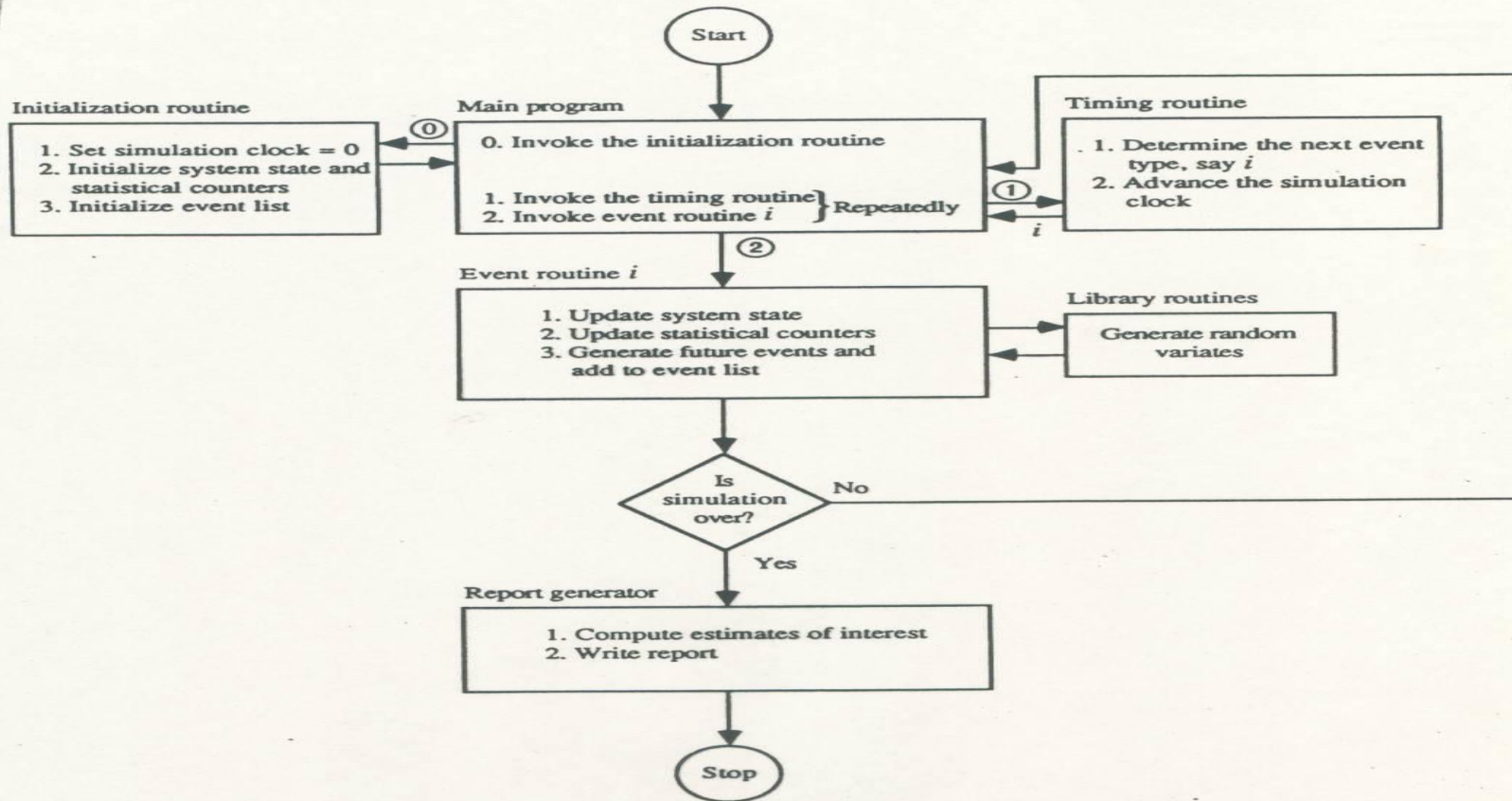
## Next-Event Time-Advance for Single-Server Queueing System

---

- In single server queueing system we define the following notation:
- $t_i$  = time of  $i$ -th customer's arrival.
- $A_i = t_i - t_{i-1}$  = Interarrival time.
- $S_i$  = service time for  $i$ -th customer.
- $D_i$  = waiting (delay) time in queue for  $i$ -th customer.
- $c_i = t_i + D_i + S_i$  = departure time for  $i$ -th customer.
- $e_i$  = time of  $i$ -th event ( $i$ -th simulation clock)

- 
- All defined parameters are random variables with known (measured) CDF generated from  $A_1, A_2, \dots$  and  $S_1, S_2, \dots$ . The following processes are performed:
    - $e_0=0$ : server idle,  $t_1$  is obtained from randomly generated  $A_1$ .
    - Simulation clock:  $e_1=t_1$ , if server idle then,  $D_1=0$ , server busy
    - $c_1$  is computed from randomly generated  $S_1$ ;  
 $c_1=S_1+t_1$
    - $t_2=t_1+A_2$
    - If  $t_2 < c_1$ , then  $e_2(\text{sim.clock})=t_2$  then:
    - $\text{Num.Custom}=\text{Num.Custom}+1=1$

- 
- If  $t_2 < c_1$ , then  $e_2(\text{sim.clock}) = t_2$  then:
    - $\text{Num.Custom} = \text{Num.Custom} + 1 = 1$
  - If  $c_1 < t_2$ , then  $e_2 = c_1$
  - $t_3 = t_2 + A_3$  again we have “if” statement:
    - If  $c_1 < t_3$ , then,  $e_3 = c_1$
  - Delay in queue:  $D_2 = c_1 - t_2$
  - $c_2 = c_1 + S_2$ ; ( $S_2$  is generated randomly)
  - $\text{Num.Custom} = \text{Num.Custom} - 1$
  - If  $t_3 < c_2$ , then,  $e_4 = t_3$
  - ... to terminate simulation



**FIGURE 1.3**  
Flow of control for the next-event time-advance approach.

customers in the facility. The server has the attribute “server status” (busy or idle), and the customers waiting in queue have the attribute “time of arrival.” (The number of customers in the queue might also be considered an attribute of the server.) Furthermore, as we shall see in Sec. 1.4, these customers in queue will be grouped together in a list.

The organization and action of a discrete-event simulation program using the next-event time-advance mechanism as depicted above is fairly typical when coding such simulations in a general-purpose programming language such as FORTRAN, Pascal, or C; it is called the *event-scheduling approach to simulation modeling*, since the times of future events are explicitly coded into the model and are scheduled to occur in the simulated future. It should be



$p_i$  is the observed (rather than expected) proportion of the time during the simulation that there were  $i$  customers in the queue. Computationally, it is easier to rewrite  $\hat{q}(n)$  using some geometric considerations. Let  $T$  be the total time during the simulation that the queue is of length  $i$ ,  $T = T_0 + T_1 + T_2 + \dots$  and  $\hat{p}_i = T_i/T(n)$ , so that we can rewrite Eq. (1.5) as

$$\hat{q}(n) = \frac{\sum_{i=0}^{\infty} iT_i}{T(n)} \quad (1.5)$$

Figure 1.5 illustrates a possible time path, or realization, of  $Q(t)$  for this system with  $n = 6$ ; ignore the shading for now. Arrivals occur at times 0.1, 3.8, 4.0, 5.6, 5.8, and 7.2. Departures (service completions) occur at times 2.4, 3.1, 3.3, 4.9, and 8.6, and the simulation ends at time  $T(6) = 8.6$ . Remember in looking at Fig. 1.5 that  $Q(t)$  does not count the customer being served, even though the queue is empty [ $Q(t) = 0$ ]; the same is true between times 3.1 and 3.3, between times 3.8 and 4.0, and between times 4.9 and 5.6. Between times 3.3 and 3.8, however, the system is empty and the server is idle, as is obviously the case between times 0 and 0.1. To compute  $\hat{q}(n)$ , we must first compute the  $T_i$ 's, which can be read off Fig. 1.5 as the (sometimes separated) intervals over which  $Q(t)$  is equal to 0

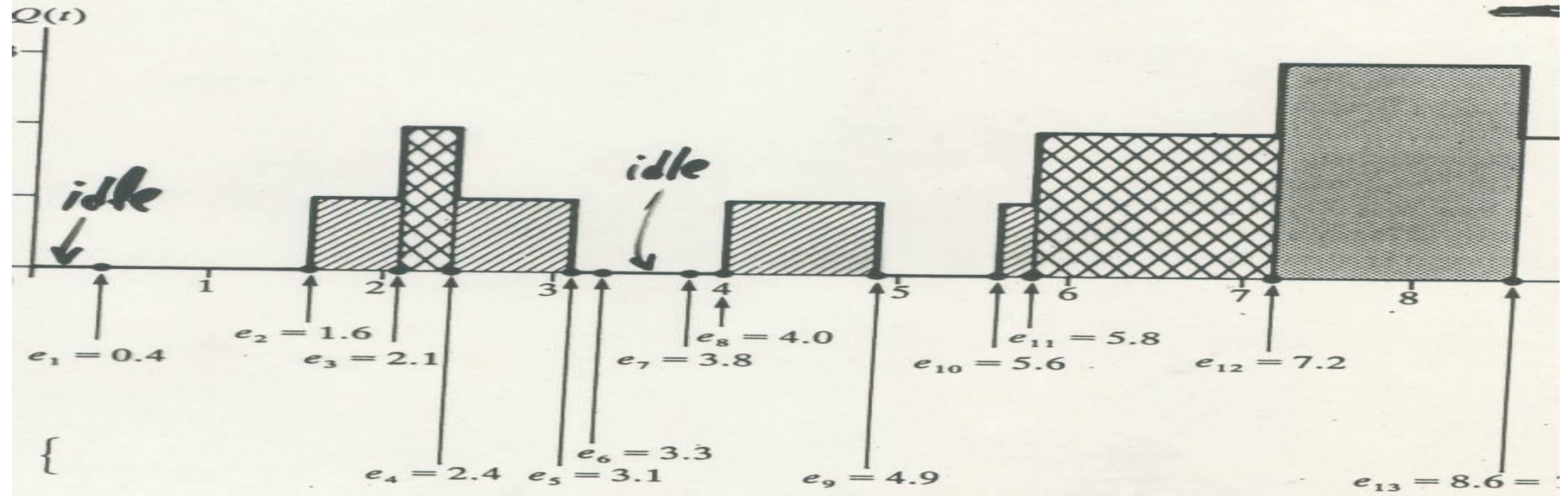
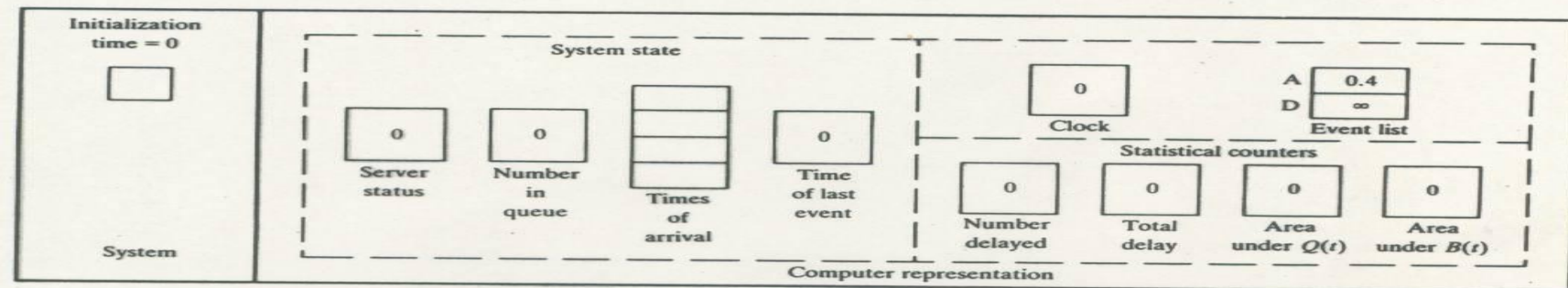


Figure 1.5: Arrival and departure times for a realization of a single-server queuing system.

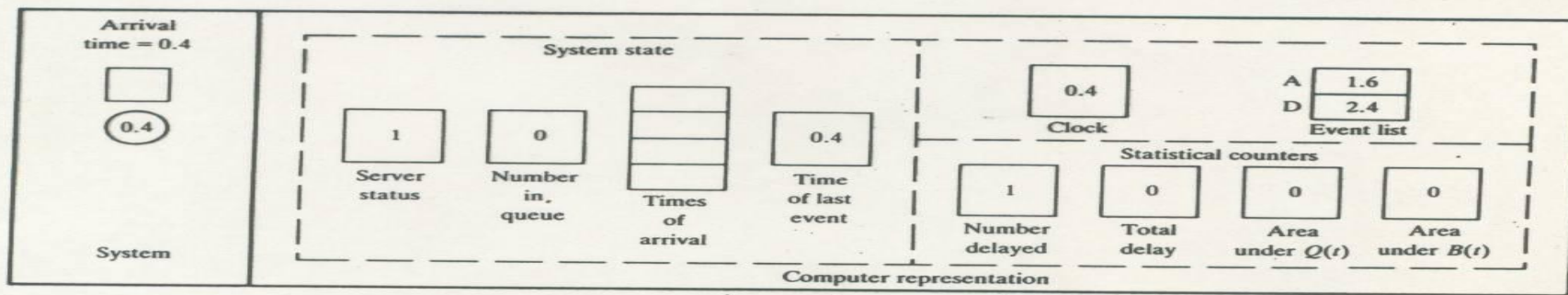
units. In an actual simulation (see Secs. 1.4.4 through 1.4.6), the  $A_i$ 's and the  $S_i$ 's would be generated from their corresponding probability distributions, as needed, during the course of the simulation. The numerical values for the  $A_i$ 's and the  $S_i$ 's given above have been artificially chosen so as to generate the same simulation realization as depicted in Figs. 1.5 and 1.6 illustrating the  $Q(t)$  and  $B(t)$  processes.

Figure 1.7 gives a snapshot of the system itself and of a computer representation of the system at each of the times  $e_0 = 0, e_1 = 0.4, \dots, e_{13} = 8.6$ . In the "system" pictures, the square represents the server, and circles represent customers; the numbers inside the customer circles are the times of their arrivals. In the "computer representation" pictures, the values of the variables shown are after all processing has been completed at that event. Our discussion will focus on how the computer representation changes at the event times.

$t = 0$ : *Initialization.* The simulation begins with the main program invoking the initialization routine. Our modeling assumption was that



(a)



(b)

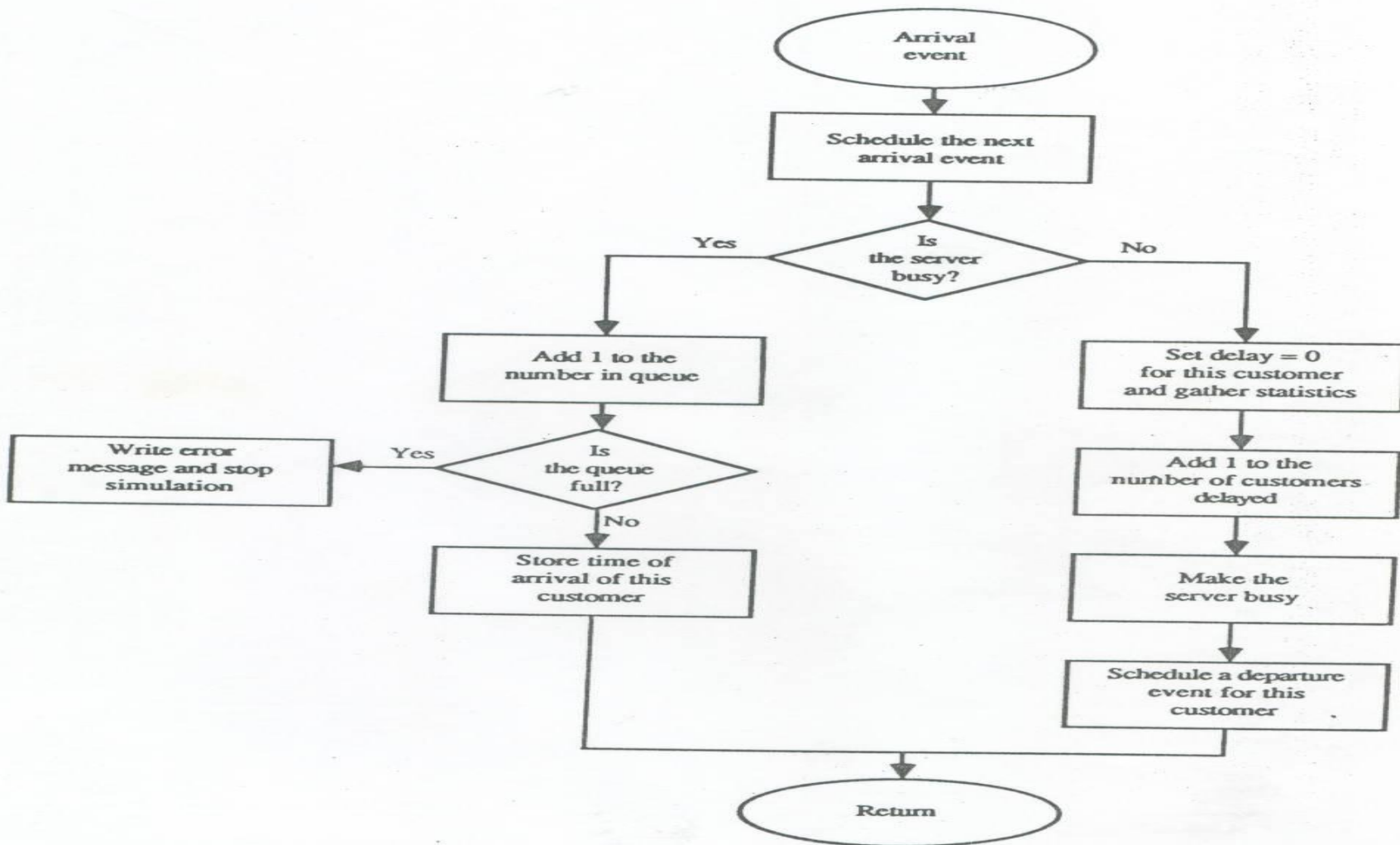
**FIGURE 1.7** Snapshots of the system and of its computer representation at time 0 and at each of the thirteen succeeding event times.



statistic, being the accumulated areas under the  $Q(t)$  and  $B(t)$  curves. The most important action, however, takes place in the routines for the events, which we number as follows:

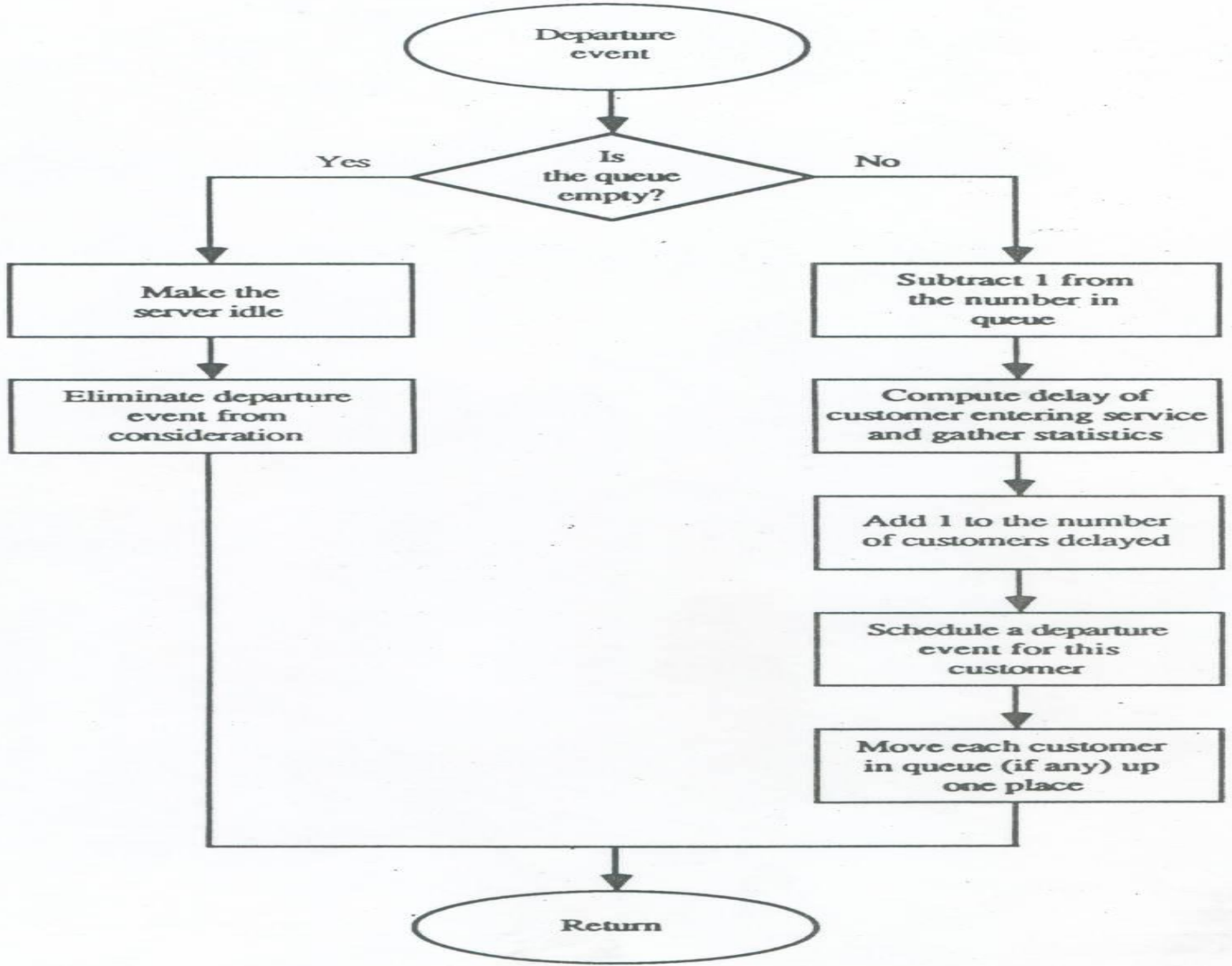
Event description	Event type
Arrival of a customer to the system	1
Departure of a customer from the system after completing service	2

As the logic of these event routines is independent of the particular language to be used, we shall discuss it here. Figure 1.8 contains a flowchart for

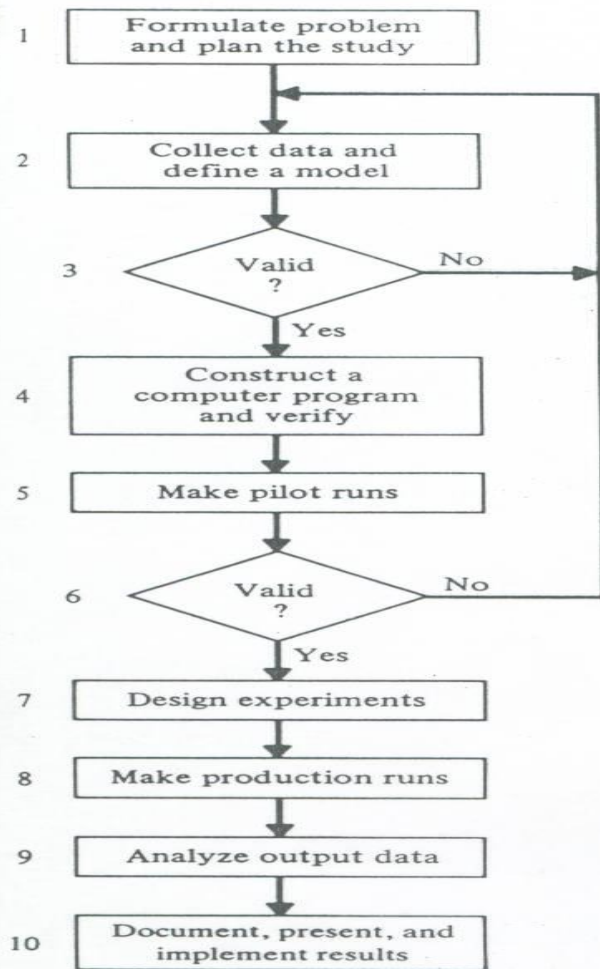




server is busy. If so, the number of customers in the queue is incremented by one, and we ask whether the storage space allocated to the queue is already full (see the code in Sec. 1.4.4, 1.4.5, or 1.4.6 for details).

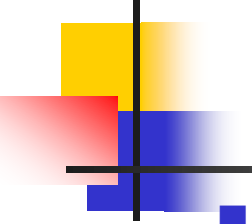


**FIGURE 1.9** Flowchart for departure routine, queuing model.



**FIGURE 1.91**  
Steps in a simulation study.

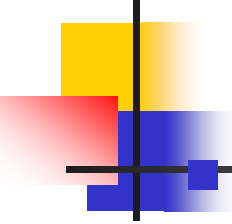
collect interarrival times and service times and use these data to specify interarrival-time and service-time distributions for use in the model. If possible, data on the performance of the system, e.g., delays in queue of customers in a bank, should be collected for validation purposes in step 6. The construction of a mathematical and logical model of a real system for a given objective is still as much an art as it is a science. Although there are few firm rules on how one should go about the modeling process, one point on which most authors agree is that it is always a good idea to start with a model that is only moderately detailed, which can later be made more sophisticated if necessary. A model should contain only enough detail to capture the essence of the system for the purposes for which the model is intended; it is not necessary to have a one-to-one correspondence between



# Components and Organization of a Discrete-Event Simulation Model

---

- 1- System state: Variables necessary to describe the system.
- 2- Simulation clock: shows the current value of simulation time.
- 3- Statistical counters: for sorting information about system.
- 4- Initialization routine: subprogram to initialize at time zero.

- 
- 5- Timing routine: subprogram that determines the next event.
  - 6- Event routine: subprogram that updates the system state.
  - 7- Library routine: subprogram used to generate random values.
  - 8- Report generator: subprogram that estimates the desired values at the end.
  - 9- Main program: subprogram that invokes the timing routine to determine the next event and transfers control and update the system state.

# Flow of control for the next- event time-advanced approach



---



## System State

- System is collection of entities.
- Entities are known by data values called attributes.
- In the single-server queueing problem, entities are the server and the customers.
- Server status (busy or idle) are attribute for server.
- Time of arrival is attribute for customer. (number of customer also could be another attribute).



## Simulation Approaches

---

- Two approaches are known for simulation:
  - 1- Event-scheduling approach, which is used in FORTRAN, Pascal, C languages.
  - 2- Process approach, describes the experience of typical entity as it flow through the system. Special-purpose simulation software is required.

# Simulation of a Single-Server Queueing System

- Even for this simple system (with one operator), we will see how simulation is complicated.
- Problem Statement:
- Inter-arrival times:  $A_i$  generated by random variable generator (IID)
- Service times:  $S_i$  also generated by random variable generator (IID)
- Simulation starts at queue-empty and server-idle
- After first arrival,  $A_1$ , server start working (not at  $t=0$  necessarily)
- Upon completing service for a customer, the server chooses a customer from queue (if any) in a first-in, first-out (FIFO) manner.
- We wish to end simulation at a fixed number ( $n$ ) of



# Measurement of the System Performance

■ Three quantities are to be estimated:

- 1- Estimation of the expected average delay  $d(n)$ . For  $n$ -customers, the average delay from a single run is:

$$\hat{d}(n) = \frac{\sum_{i=1}^n D_i}{n}$$

- We should count  $D_j=0$
- $d(n)$  explains system performance from the the view point of the customer.



# Measurement of the System Performance

2- The expected average number of customers in the queue  $q(n)$ :

$$q(n) = \sum_{i=0}^{\infty} ip_i$$

- Where,  $p_i$  is probability that  $i$  customers are in queue.  $p_i$  can be observed by the proportion of the time ( $T_i$ ) that  $i$  customers were in the que

$$q(n) = \frac{\sum_{i=0}^{\infty} iT_i}{T(n)}$$

- Where,  $T(n)$  is total summation time.
- $q(n)$  explains system performance from

## Example of Average Number of Customers in Queue

In Fig.1.5:

- $T_0 = (1.6 - 0.0) + (4.0 - 3.1) + (5.6 - 4.9) = 3.2$
- $T_1 = (2.1 - 1.6) + (3.1 - 2.4) + (4.9 - 4.0) + (5.8 - 5.6) = 2.3$
- $T_2 = (2.4 - 2.1) + (7.2 - 5.8) = 1.7$
- $T_3 = (8.6 - 7.2) = 1.4$
- $\sum_i T_i = (0 \times 3.2) + (1 \times 2.3) + (2 \times 1.7) + (3 \times 1.4) = 9.9$
- $q(6) = 9.9 / 8.6 = 1.15$
- $\sum_i T_i = \int Q(t) dt$

# Measurement of the System Performance

- 3- Server efficiency:
- Expectation of utilization of the server:  
 $u(t)$
- $B(t)=1$  if server is busy
- $B(t)=0$  if server is idle
- In example (Fig.1.6)
- $u(t)=[(3.3-0.4)+(8.6-3.8)]/8.6=0.9=\%90$
- $u(t)=\int B(t)dt/ T(n)$



## M/M/1 Queue

---

- Definition:
- The single server queue with exponential interarrival and service times is called the M/M/1 queue

## "C" Programming for MM1

- `*/C Program for mm1 */`
- `#include<stdio.h>`
- ~~`#include<stdlib.h>`~~
- `#include<math.h>`
- `#include<time.h>`
- `#define NUM 1000`
- `#define MAXIDOL 3600;`
- `#define MAXSERVT 1800;`
- `/*`
- `* MAXIDOL/2 = expectation of interarrival time`
- `* MAXSERVT/2 = expectation of serving time`
- `* t=time of arrival of the ith customer(t0=0)`
- `* A=interarrival time`
- `* S=time that server actually spends serving ith customer`
- `* D=delay in queue of ith customer`
- `* c=t+D+S=time that ith customer completes service and departs`
- `* e=time of occurrence of ith event of any type`
- `*/`

- int main()
- {
- int S[NUM],t[NUM];
- t[0]=rand()%MAXSERVT;
- int A[NUM],D[NUM],c[NUM],e[NUM],i,j,stop,stoptime;
- float tsum,dtave;
- /\*make random variable\*/
- srand((unsigned) time(NULL));
- S[0]=rand()%MAXSERVT;
- for(i=1;i<NUM;i++){
- A[i]=rand()%MAXIDOL;
- t[i]=t[i-1]+A[i];
- S[i]=rand()%MAXSERVT;
- if(t[i]>=60\*60\*9){
- stop=i-1;
- stoptime=t[i-1];
- i=NUM;
- }
- }

}  
}

```
for(i=0;i<=stop;i++){  
    printf("t[%2d] %2d:%2d:%2d ¥n",i,t[i]/(60*60),  
          (t[i]%(60*60))/60,t[i]%60);  
}
```

```
A[0]=0;
```

```
D[0]=0;
```

```
c[0]=0;
```

```
e[0]=0;
```

```
for(i=1;i<=stop;i++){
```

```
    c[i-1]=t[i-1]+S[i-1]+D[i-1];
```

```
    if(c[i-1]>t[i])
```

```
        D[i]=c[i-1]-t[i];
```

```
    else
```

```
        D[i]=0;
```

```
    }
```

```
for(i=0;i<=stop;i++){
```

```
    printf("D[%2d] %2d:%2d:%2d
```



```

    ¥n",i,D[i]/(60*60),
                                     (D[i%60*60))/60,D[i%60);
    }
    printf("n=%d
stoptime=%2d:%2d:%2d¥n",stop+1,stoptime/3600,(stoptime%3600)/60,
stoptime%60);
    for(i=0;i<=stop;i++){
        tsum=tsum+D[i];
    }
    dtave=tsum/(stop+1);

printf("average=%2d:%2d:%2d¥n",(int)dtave/3600,((int)dtave%3600)/60,
((int)dtave%60));

int Dnum[stop];
for(i=0;i<stop;i++)
    Dnum[i]=0;
for(j=1;j<stop-1;j++){

```



- [asharif@nirai ~]\$ less mm1.c
- [asharif@nirai ~]\$ gcc mm1.c
- [asharif@nirai ~]\$ ./a.out
- t[ 0] 0:29:43
- t[ 1] 1:16: 9
- t[ 2] 1:37:49
- t[ 3] 2:11:19
- t[ 4] 2:23:49
- t[ 5] 3:16:48
- t[ 6] 3:40:46
- t[ 7] 4:28:47
- t[ 8] 4:56:10
- t[ 9] 5:43:49
- t[10] 6: 4:13

- t[11] 6:18:29
- t[12] 7:17:31
- t[13] 8: 1:56
- t[14] 8: 8:36
- t[15] 8:59:46
- D[ 0] 0: 0: 0
- D[ 1] 0: 0: 0
- D[ 2] 0: 0: 0
- D[ 3] 0: 0: 0
- D[ 4] 0: 0: 0
- D[ 5] 0: 0: 0
- D[ 6] 0: 0: 0
- D[ 7] 0: 0: 0
- D[ 8] 0: 0: 0
- D[ 9] 0: 0: 0
- D[10] 0: 2: 2
- D[11] 0: 8:50
- D[12] 0: 0: 0
- D[13] 0: 0: 0
- D[14] 0: 0: 0
- D[15] 0: 0: 0

- n=16 stoptime= 8:59:46
  - average= 0: 0:40
  - queue 1 = 2
- 
- [asharif@nirai ~]\$



## Step in a Simulation Study

---

- 1. Formulate problem and plan the study.
- 2. Collect data and define a model.
- 3. Valid
- 4. Construct a computer program and verify.
- 5. Make pilot runs.
- 6. Valid
- 7. Design experiments.
- 8. Make production runs.
- 9. Analyze output data.



## Continuous simulation

---

- ACSL, CSSL-IV, simulation languages have been designed for continuous simulation.
- Discrete-event simulation language:  
SIMAN, SIMSCRIPT II.5, SLAM II also have continuous modeling capabilities.



## Example of a Continuous Model: Predator-prey

- Is a continuous model of competition between two populations (Predator & prey)
- $x(t)$ : prey number
- $y(t)$ : predator number
- $dx/dt = r \cdot x(t) - a \cdot x(t) \cdot y(t)$
- $dy/dt = -s \cdot y(t) + b \cdot x(t) \cdot y(t)$
- For some  $T > 0$ :
- $x(t+nT) = x(t)$  and
- $y(t+nT) = y(t)$





## Monte Carlo Simulation

---

- *Monte Carlo* simulation is a scheme in which random numbers are used. Sometimes it is restricted and referred to static cases using random numbers. The name was originated during World War II for developing atomic bomb.



# Deterministic example by Monte Carlo Simulation

---

- $I = \int g(x) dx$

$$Y = (b-a) g(X)$$

$$E(Y) = E[(b-a) g(X)] = (b-a) E[g(X)]$$

$$= (b-a) \int g(x) f(x) dx$$

$$= (b-a) \int g(x) dx / (b-a) = I$$

Where  $f(x) = 1/(b-a)$

$$I = E[Y(n)] = (b-a) \Sigma g(X) / n$$