Concept of discrete-event-systems

ideas of * events

* event notices

* event list

e.g. simple single-server queue

objective: how a discrete-event simulation operates

Introduction

e.g.

Consider a system of checkout lanes at a supermarket

Suppose there are six lanes, with the following characteristics.

1) A study shows that time intervals at the checkout process are NOT exponentially distributed
2) Payments are carried out in cash, cheque, creditcard, debit Card
3) Length of time ~~beeing~~ checking out depends on number of items, method of payment and speed of operator.
4) Two express lanes are for customers with <12 items, and cash payments only.
5) Waiting customers will jockey to a shorter line when their line is at least two customers longer than the shorter line.

It is not possible to represent such a model as a Markov chain or a similar model to that developed before.

Therefore we need a more powerful modeling framework that we can use to describe such models.

other examples are

* manufacturing plants
* communication networks
* transportation systems
* ... systems that involve service, queuing and processing

We therefore need a common general modeling framework that can be used to describe many operational systems.

Static and Dynamic Model Description

consider the checkout systems

To describe such a system we need to consider the following

1) The components involved
2) The way the components interact together.

Static model

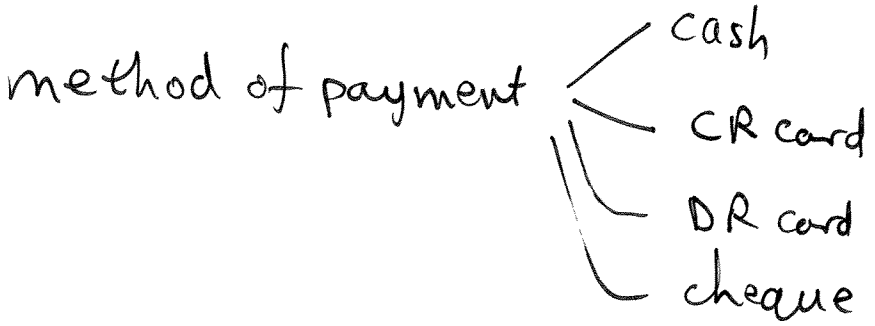
This provides a list of all components in the system together with all relevant information about them.

e.g. 1) There are six checkout lanes

Lane is either express / regular

Lane has a queue to hold waiting customers

2) Customers have two characteristics
   a) number of items
   b) method of payment



Dynamic model

Describes the interaction between components.
Provides a set of rules telling how the components interact as time advances.

e.g.

1) time between customer arrivals is uniformly distributed between 10secs & 60 secs.
2) An arriving customer selects the checkout lane containing the smallest no. of customers.
3) Customers are served on First come First served basis.
4) Conditions under which customer jockays to another queue.

State of a system

This is a collection of variables containing all information necessary to operate the model over time.

Change in a state variable implies a change in the system.

e.g. simple queueing system

1 queue

1 server

The state of the system can be represented using two variables

Q → the no of customers in the queue

S → status of the server busy/idle

Events and state changes

The system state changes only at discrete points in time.

These points mark the occurence of an event.

Therefore the state remains constant between these points.

e.g. in a simple queueing system the state when

1. a customer arrives
2. a customer begins service
3. a customer completes service

] These are called events

An <u>event</u> is defined to be any occurrence that causes an instantaneous change in the system state.

Each type of an event has an associated set of instructions, called an event routine, that specifies exactly how the system state changes when an event of the given type occurs.

Event Scheduling

A mechanism must be provided to increment time by a variable amount between events.

We must also be able to sequence events according to the time of occurence and apply the associated event routine to make the changes in the state.

To do this create an event notice for each event when its future time is determined.

At two items are required on this event notice

1. time of occurence
2. type of event

All the notices are placed in an event list which is ordered by time of occurence.

Then the next event to occur is at the top of the list.

The process of

1. creating an event notice
2. recording the necessary information on the event
3. placing it in the event list

is called event scheduling.

System clock

This is a special system variable that holds the current system time.

The process of

1. selecting the first event notice
2. system clock = time of this event
3. executing the event routine
4. discarding the event notice

is called the timing routine

Developing the Model

The following need to be determined

1. What variables or components are needed to adequately represent the system state,
2. What events are needed to represent the system changes,
3. What state changes occur within each event, including the details of which events are scheduled and when they are scheduled

Most of the time an iterative process is used.

So first establish a set of variables ~~and~~ event types and event routines.

When additional information is needed modify the variables or components and update the routines.

It is easy to see that there is no unique model for a given system.

A Generic Simulation Language [GSL]

We will develop a simple GSL that can be used to describe the system components and the changes that occur in the event routines.

General purpose Languages (e.g. C, Java, Pascal) have libraries that allow the development of discrete event simulation. So you can try these if you want (But you will not get any & marks here!)

GSL will allow us to focus on simulation concepts without being involved with the syntax of a computer Language. for example data types are left out on purpose.

We will assume that GSL has the following.

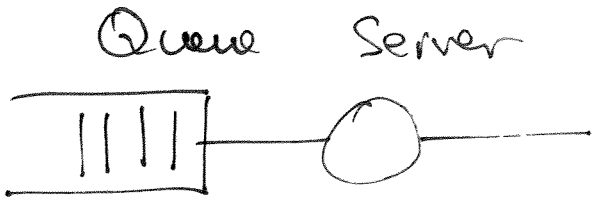
General Programming Language Features

Constants
Variables
Arrays of variables
Arithmetic and logical operators
Assignment statement A:=B+C/D
Commands or procedures to read in or write out constants and variable values
Looping statements: While..Do, For..do
Conditional statements: If...Then..Else
Procedures with parameters
Functions with parameters

Event Definition and Scheduling Facilities

Event List
System clock: NOW
Timing routine: simulate
Event routines
Commands to:
Schedule an event: schedule [event type] at time [T]
Find an event notice: find event
Cancel an event: cancel event

Example: A Single-Server Queue



service times
→ exponential distribution
mean = $1/\mu$

inter-arrival times
→ exponential distribution
mean = $1/\lambda$

λ = arrival rate
μ = service rate

Assume $\lambda < \mu$

∴ system has a stationary distribution

We <u>would like</u> to estimate the mean number of customers in queue when the system is operating in steady state.

State Variables

All customers are identical
- No billing
- No need to track them individually

So we will simply keep a count of the number of customers in the queue.

We also need to know if the server is busy or idle

Then define

Q : no of customers [0, 1, 2, 3, ...]

S : status of server [idle/busy 0/1]

Event Routines

Define event types

1. arrival
2. begin service
3. end service

The event routines are the following

```
Event routine arrival
Schedule arrival event at time NOW + interarrival time
Q := Q + 1
IF S := idle THEN schedule begin service event at time NOW

Event routine begin service
Q := Q - 1
S := busy
Schedule end service event at time NOW + service time

Event routine end service
S := idle
If Q > 0
   THEN schedule begin service event at time NOW
```

In GSL 'NOW' is a variable that contains the current system time

'interarrival time' is a function that returns randomly sampled value for iat

same for 'service time'.

Statements like

'Schedule [event-type] at time[T]'

mean

place an event notice with event time 'T' and event type [event-type] on the event list.

Initialize

Before starting the simulation we initialize, for example as follows

```
Routine Initialize
Q := 0
S := idle
Schedule arrival event at time NOW
```

Simulate

The main program is then

Initialize
Simulate

Dry Run!

At the start we have

State Variables

Q	*S*	*NOW*

Event List

Number	Time	Event Type

After initialisation we have

State Variables

Q	*S*	*NOW*
0	*idle*	0.0

Event List

Number	*Time*	*Event Type*
→ 1	0.0	Arrival

Then control is turned over to simulate

```
Simulate
WHILE the event list is not empty and stop signal has not been
      received DO:
   Update the clock to the time on the first event notice.
   Execute the event routine for the type of event on the first
   event notice.
Discard the first event notice.
```

For this dry run we will use the following random numbers.

TABLE 5.1 Randomly sampled observations for interarrival and service times for the single-server queue model

Interarrival times	.5	1.4	.1	1.7	.8	.1	.1	2.6	3.5	.2
Service times	1.0	.2	.8	.8	.4	.5	.4	2.6	.5	.0

At this point the timing routine takes control.

There is one event → arrival event

→ execute arrival routine

Q := Q + 1

schedule arrival event at

NOW + iat

0.0 + 0.5 = 0.5

schedule begin_service at

NOW = 0.0

At this point state variables and events are:

System State

Q
1

S
idle

NOW
0.0

Event List

Number	Time	Event Type
→ 3	0.0	Begin service
2	.5	Arrival

event notice no.1 is discarded

Now first event is Begin-service

Q: Q - 1

S: busy

schedule end-service at

NOW + 1.0 = 1.0

and we update SV & EL

System State

Q
0

S
busy

NOW
0.0

Event List

Number	Time	Event Type
→ 2	.5	Arrival
4	1.0	End service

Next Event is an arrival but the server is busy. So we have

System State

Q
1

S
busy

NOW
.5

Event List

Number	Time	Event Type
→ 4	1.0	End service
5	1.9	Arrival

Next event is an end-of service updated SV & EL :

System State

Q
0

S
busy

NOW
1.0

Event List

Number	Time	Event Type
→ 6	1.0	Begin service
5	1.9	Arrival

And the next update is

System State

Q
0

S
busy

NOW
1.0

Event List

Number	Time	Event Type
→ 7	1.2	End service
5	1.9	Arrival

We need a way to stop the simulation for example

if we wanted to make the simulation stop as soon as the first event on or after time 10.0 is executed, then we could replace the statement:

Schedule arrival event at time NOW + interarrival time.

with the statement:

IF *NOW* < 10.0
THEN Schedule *arrival* event at time *NOW* + interarrival time.

Summary of first 20 events

Simulation of an M/M/1 Queueing System:
Arrival Rate 0.700
Service Rate 1.000

Time	Event	Action(s)	
		Run trace for 20 events.	
		System initialized to have 0 customers.	
0.000	Arrival	Schedule Arrival event at time	0.000
		---------- Start event ----------	
		Schedule Arrival event at time	1.787
		Add customer to queue. Length is 1	
		Server status is idle.	
0.000	Begin Serv	Schedule Begin Serv event at time	0.000
		---------- Start event ----------	
		Take customer from queue. Length is 0	
		Server is now busy.	
0.124	End Serv	Schedule End Serv event at time	0.124
		---------- Start event ----------	
1.787	Arrival	Queue length is 0. Server is idle.	
		---------- Start event ----------	
		Schedule Arrival event at time	2.602
		Add customer to queue. Length is 1	
		Server status is idle.	
1.787	Begin Serv	Schedule Begin Serv event at time	1.787
		---------- Start event ----------	
		Take customer from queue. Length is 0	
		Server is now busy.	
2.602	Arrival	Schedule End Serv event at time	4.492
		---------- Start event ----------	
		Schedule Arrival event at time	3.708
		Add customer to queue. Length is 1	
3.708	Arrival	Server status is busy.	
		---------- Start event ----------	
		Schedule Arrival event at time	4.625
		Add customer to queue. Length is 2	
4.492	End Serv	Server status is busy.	
		---------- Start event ----------	
		Queue length is 2. Server is idle.	
4.492	Begin Serv	Schedule Begin Serv event at time	4.492
		---------- Start event ----------	
		Take customer from queue. Length is 1	
		Server is now busy.	
		Schedule End Serv event at time	8.460

Time	Event	Action(s)	
4.625	Arrival	---------- Start event ----------	
		Schedule Arrival event at time	5.002
		Add customer to queue. Length is 2	
		Server status is busy.	
5.002	Arrival	---------- Start event ----------	
		Schedule Arrival event at time	7.444
		Add customer to queue. Length is 3	
		Server status is busy.	
7.444	Arrival	---------- Start event ----------	
		Schedule Arrival event at time	8.003
		Add customer to queue. Length is 4	
		Server status is busy.	
8.003	Arrival	---------- Start event ----------	
		Schedule Arrival event at time	9.250
		Add customer to queue. Length is 5	
		Server status is busy.	
8.460	End Serv	---------- Start event ----------	
		Queue length is 5. Server is idle.	
8.460	Begin Serv	Schedule Begin Serv event at time	8.460
		---------- Start event ----------	
		Take customer from queue. Length is 4	
		Server is now busy.	
8.557	End Serv	Schedule End Serv event at time	8.557
		---------- Start event ----------	
		Queue length is 4. Server is idle.	
8.557	Begin Serv	Schedule Begin Serv event at time	8.557
		---------- Start event ----------	
		Take customer from queue. Length is 3	
		Server is now busy.	
8.715	End Serv	Schedule End Serv event at time	8.715
		---------- Start event ----------	
		Queue length is 3. Server is idle.	
8.715	Begin Serv	Schedule Begin Serv event at time	8.715
		---------- Start event ----------	
		Take customer from queue. Length is 2	
		Server is now busy.	
9.125	End Serv	Schedule End Serv event at time	9.125
		---------- Start event ----------	
		Queue length is 2. Server is idle.	
		Schedule Begin Serv event at time	9.125

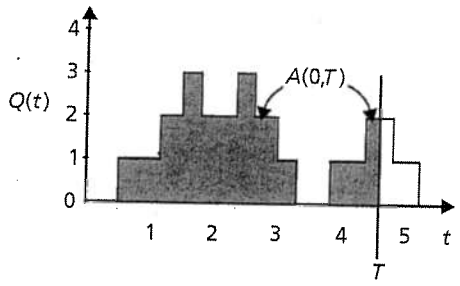
Calculating the Mean Number in the Queue

This is after all our quest!

The number of customers in the queue as a function of time = Q

= Q(t) = queue length at time t

FIGURE 5.12
Plot of $Q(t)$ with $A(0,T)$



Average no of customers in queue

$$= \frac{\text{Area under graph}}{T}$$

$$\mu = \lim_{T \to \infty} \frac{A(0,T)}{T}$$

To eliminate the transient part we remove the first batch of results.

$$\mu = \frac{A(t_s, t_s+T)}{T}$$

The area = $\sum$ rectangles

or

$$A(t_s, t_s+T) = \sum_{k=1}^{k} Q_k (t_k - t_{k-1})$$

where

Q_k = number in queue between $(k-1)^{th}$ and k^{th} events

t_k = time of k^{th} event

Events are numbered from $t = t_s$

We therefore have to <u>save</u> the time of the previous event. = tprev

Then compute

$$A = Q\,(\text{NOW} - t_{prev})$$

and add A to Ac, which is the cumulative sum.

Then $\mu = \frac{A_c}{T}$

To compute a confidence interval we can use the batch means method.

Define batches in time segments

Let B = number of batches

L = length of batch in time units

Then the first batch mean is

$$\bar{X}_1 = \frac{A(t_s, t_s+L)}{L}$$

and

$$\bar{X}_2 = \frac{A(t_s+L, t_s+2L)}{L}$$

and last one

$$\bar{X}_B = \frac{A(t_s+(B-1)L, t_s+BL)}{L}$$

Also $T = BL$

and average of Batch Means is

$$\bar{\bar{X}} = \frac{1}{B} \sum_{K=0}^{B} \bar{X}_K$$

$$= \text{time average from } t_s \text{ to } t_s + T$$

To compute variance treat $\bar{X}_1, \bar{X}_2, \ldots \bar{X}_B$ as independent observations, and

$$s^2 = \frac{1}{B-1} \sum_{i=1}^{B} (\bar{X}_i - \bar{\bar{X}})^2$$

Then the confidence interval can be calculated, for example from the t-distribution as

$$\bar{\bar{X}} \pm t_{B-1, \alpha/2} \frac{s}{\sqrt{B}}$$

We now have to define data collection routines in our GSL code.

FIGURE 5.13
Event routines for single-server queue with data collection

```
Initialize
  Q := 0
  S := idle
  Schedule arrival event at time NOW
  C := 0
  AREA := 0.0
  Schedule data-collection event at time NOW + TRANS

Event routine data collection
  AREA := AREA + Q * (NOW - OLDT)
  OLDT := NOW
  IF C > 0 THEN BM[C] := AREA/L
  IF C < B THEN
    AREA := 0
    C := C + 1
    Schedule data-collection event at time NOW + L
  ELSE stop simulation

Event routine arrival
  AREA := AREA + Q * (NOW - OLDT)
  OLDT := NOW
  Schedule arrival event at time NOW + interarrival time
  Q := Q + 1
  IF S := idle THEN schedule the begin service event at time
    NOW

Event routine begin service
  AREA := AREA + Q * (NOW - OLDT)
  OLDT := NOW
  Q := Q - 1
  S := busy
  Schedule end service event at time NOW + service time

Event routine end service
  AREA := AREA + Q * (NOW - OLDT)
  OLDT := NOW
  S := idle
  IF Q > 0 THEN schedule begin service event at time NOW
```

Notes

1) Data collection event is new
This occurs at the <u>end</u> and <u>start</u> of a batch

2) L = length of each batch

B = number of batches

TRANS = T_s

AREA = A_c

C = cumulative batch count

OLDT = $t_{previous}$

BM = an array of B elements to contain $\bar{X}_k$

3) Modification to Initialise

Data collection at NOW + TRANS scheduled

4) The Data collection routine is used to:

* determine if transient period is over
* Store $\bar{X}_k$ in BM
* Stop the simulation