Chapter 6
Simulation Language

Basic concept of simulation Software

Simulation software is a program that allows the user to observe an operation through simulation without actually performing that operation. Simulation software is used widely to design equipment so that the final product will be as close to design specs as possible without expensive in process modification.

There are essentially three types of languages one can use for simulation:.

a) General purpose programming language

b) Simulation programming language

c) Simulation tools or environment or software

General purpose programming language such as Java, C++, C can be used to developed the simulation model of real system as a computer program and can be used for the experimentation purpose. It may be the difficult and time consuming task since the general purpose programming language may not support directly the tools and function required by a particular simulation model. In such a case, we have to use specially designed language for simulation.

These are languages used for many applications. There are a number of advantages to implementing simulations in general languages : they tend to be very fast, there are few limitations on what can be done, programmers are easily found, and simulations can be done on a number of different computers (due to standardization of language). The disadvantage of this approach is cost. Simulations based on general languages take knowledge of simulation implementation and are generally very large, intricate programs. The time needed to design, code, and verify such a system may be overwhelming.

The second class of languages is simulation specific, and includes GPSS, SIMAN, etc. These languages take most of the work out of creating a simulation. All these systems can do all the queue manipulation needed in a single line. The main disadvantages of these languages are in their limited domain. It is difficult, but not impossible, to create a financial portfolio planner in GPSS. It is just not designed for it. On the other hand, another specialized simulation language can easily handle it within a 123 spreadsheet. It is also difficult to use other programs in the simulation. Imagine trying to put in a network optimizer in a transportation simulator if you are required to use SIMAN. Although there is the ability to link in your own FORTRAN programs, the methods are cumbersome and inefficient.

Third, there are simulation environment or IDE such as **Arena, AutoMod** etc. these products includes common characteristics such as graphical user interface and an environment that supports all aspects of a simulation study.

Some other examples are :

ACSL, APROS, ARTIFEX, C++ SIM, CSIM, CallSim, FluidFlow, GPSS, Gepasi, JavSim, MJX, MedModel, Mesquite, Multiverse, NETWORK, OPNET Modeler, POSES++, Simulat8, Powersim, QUEST, REAL, SHIFT, SIMPLE++, SIMSCRIPT, SLAM, SMPL, SimBank, SimPlusPlus, TIERRA, Witness, SIMNON, VISSIM, and Javasim.

Specific Simulation Programming Language

The overall simulation language can be categorized into two broad classes: discrete system simulation language and continuous system simulation language.

Continuous system simulation language:

These languages use the familiar statement type of input for digital computer allowing a program to be programmed directly from the equation of mathematical model, rather than requiring the equations to be broken into o functional element.

CSSL includes a variety of algebraic and logical expressions to describe the relation between variables. Several implementation of CSSL have been published. One particular CSSL that illustrate the nature of these language is the Continuous system modeling program (CSMP, version 3) i.e. CSMP III

Discrete simulation language

A number of programming languages have been developed to simplify the task of writing discrete system simulation program. These programs include a language with which to describe the system and a programming system that will establish a system image and execute a simulation algorithm. Each language is based on set of concept used of describing system. The word world-view has come to be used to describe this aspect of simulation program. The user of the program must learn the world view of particular language he is using and be able to describe the system in those terms.

Here in our course we will discuss the GPSS : A discrete system simulation language.

Introduction to GPSS:

The system to be simulated in GPSS is described as block diagram in which the blocks represents the activities and line joining the block indicated the sequence in which the activities can be executed.

The use of block to describe the system being simulated of course, make easy and familiar to understand what kind of system we are going to simulate and what are the essential or important part of the system are included in our simulation model.

The GPSS defines a set of 48 specific block types, each of which represents a characteristics action of a system. The program user must draw a block diagram of the system using only these blocks and then write the corresponding code for each block.

GPSS block

Each block type is given a name that is descriptive of block action and is represented by a particular symbol.

A GPSS block diagram can consists of many blocks up to some limit prescribed by the program.

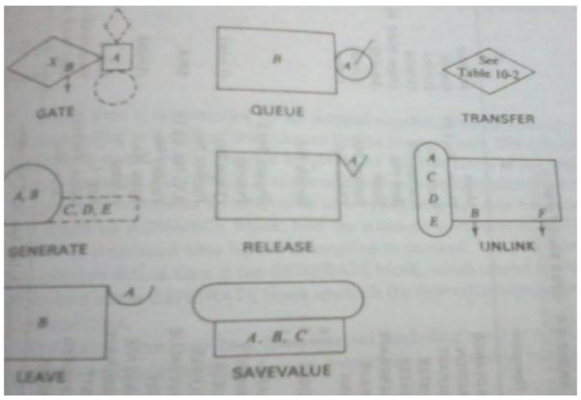
An identification number called location is given to each block.

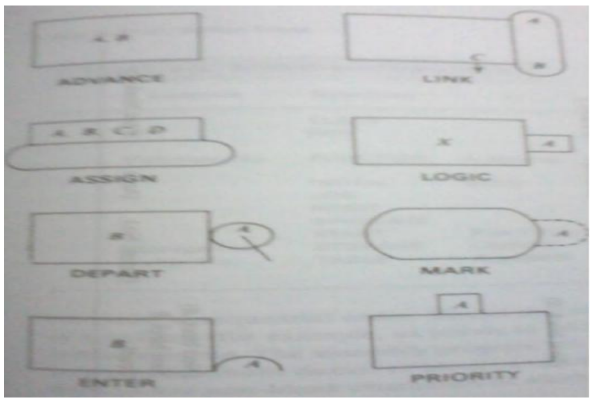
Transections

There are entities moving throughout the system. For example, a communication system is concerned with the movement massage, a road transportation system with motor vehicles. In simulation, these entities are called transactions.

The location is assigned automatically by an assembly program within GPSS so that when a problem is coded the blocks are listed in sequential orders. The assembly program will associate with the name with appropriate location. The symbolic names of blocks and other entities of the program must be from 3 to 5 non blank characters of which first three must be letters.

Pictorial representation of block





Action Times

Clock time is represented by an integer number. The unit of time is not specifically stated but all times are expressed in terms of same unit.

The program computes an interval of time called an action time for each transection as it enters an ADVANCE block and the transaction remains at the block for this interval of time before attempting to proceed.

An action time is defined by giving a mean and modifier as the A and B fields for the block. The C field however can be used to specify an offset time as the time when the first transaction will arrive. The field D is used to specify a limit to the total no of transaction that will come from the block.

The E field determines the priority of the transactions.

Succession of Events

The program maintains records of when each transaction is due to move. It proceeds by completing all movements that are scheduled for execution ata a particular time.

Normally a transaction spends no time at a block other than at an ADVANCE block.

Choice of path

The transfer block allows some location other than the next sequential location to be selected. The choice is normally between two clocks referred to as next blocks A and B. the method used for choosing is indicated by a selection factor in field of the TRANSFER block. Next block A and B are placed in field B and C respectively.

Example Program: Simulation of manufacturing shop

To illustrate feature of the GPSS described so far, we consider a simple example here. A machine tool in manufacturing shop is turning out parts at a rate of one every 5 minutes. As they are finished, the parts go to an inspection, who takes 4+-3 minutes to examine each one and reject about 10% of the parts.

Here each parts will be represented by one transition and he time unit selected fo the problem will be 1 minutes.

The block diagram representing the system is shown in figure.

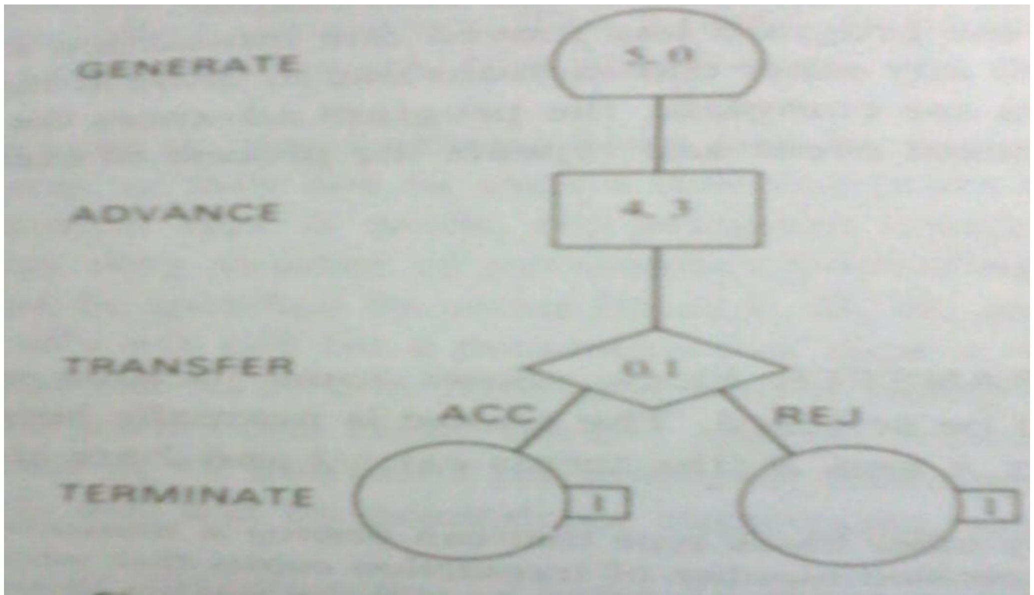


Figure 9-2. Manufacturing shop - model 1.

Here a GENERATE block is used to represent the output of the machine by creating one transaction.

An ADVANCE block with a mean of 4 and modifier of 3 is used to represent inspector. The time spent on inspection will be any one of the values 1, 2, 3, 4, 5, 6 or 7. With equal probability given t each value.

Upon completion of the inspection, transactions go to TRANSFER block with selection factor of 0.1 so that 90% of the part of to next location called ACC, to represent acceptance part ad 10% got to another location called REJ to represent rejected parts.

Since there is no further interest in the history of parts in this simulation, both location reached from the TRANSFER blocks to TERMINATE blocks.

The equivalent program code (in GPSS source code)

```
                  *manufacturing shop simulation*
        GENERATE       5               ; create parts
        ADVANCE        4,3             ;inspects
        TRANSFER       .1, ACC, REJ    ; Selects or rejects
ACC     TERMINATE      1               ; Accepted
REJ     TERMINATE      1               ; Rejected
        START          1000
```

Introduction to CSMP III

A CSMP III program is constructed form three general types of statement.

1) Structural statements which defines the model. They consist of FORTARN like statement and functional block designed for operations that frequently occurs in a model definitions.
2) Data statements which assign numerical value to parameters, constant and intial conditions.
3) Control statements which specify options in the assembly and execution of program and choice of inputs.

Structural Statements
Structural statement can make use of the operation of addition, subtraction, multiplication, division and exponentiation, using the same notation and rule as are used in FORTRAN.
If the model include the equation

$$X = \frac{6Y}{W} + (Z-2)^2$$

Then the following statement would be used

X=6.0*Y/W+(Z-2)**2.0

There are many functional block which in addition to provide operation specific to simulation. Some of them are the exponential function, trigonometric function and function for taking maximum values.
Functional Block in CSMP III

The following is the list of functional blocks available in CSMP III

S.N.	General Form	Functions
1.	Y=INTGRL(IC,X) Y(0)=IC (Name: Integrator)	$Y = \int_0^T X dt + IC$

2.	Y=LIMIT(P1,P2,X)
	(Name: limiter)
		Y=P1, X<P1
		Y=P2, X>P2
		Y=X, P1<=X<=P2

3.	Y=STEP(P)
	(Name: step function)
		Y=0, T<P
		Y=1, T>=P

4.	Y=EXP(X) (Name: Exponential)	Y=eX
5.	Y=ALOG(X) (Name: natural logarithm)	Y=log(X)
6.	Y=SIN(X) *(Name: Trigonometric Sine)*	Y=sinX
7.	Y=COS(X) (Name: Trigonometric Cosine)	Y=cosX
8.	Y=SQRT(X) *(Name: Square root of X)*	$Y=X^{1/2}$
9.	Y=ABS(X) *(Name: Absolute value of X)*	Y=|X|
10.	Y=AMAX1(X1,X2,……..Xn) (Name: Largest value among N	Y=max(x1,x2,…..xn)

	real values)	
11.	Y=AMIN(X1,X2,…………..Xn)\ (Name: Minimum value among N real values)	Y=min(x1,x2,…..xn)

Data statements:

Data statements are used to set the initial values to the model parameter. For example one data statement called INCON can be used to set the initial value of integration function block.

Control Statements

Among the control statement, TIMER is one of the control statements which specify certain time interval. For adequate accuracy, it should be small in relation to the rate at which variable change values. The following is an example

TIMER DELT=0.005, FINTIM=1.5, PRDEL=0.1

The item specified are

DELT→ integration interval

FINTIM→Finish time

PRDEL→Interval, at which to print result.

If printed output is required, control statements with PRINT and PRTPLT are used followed by the names of variables to form the outputs.

The set of structural, data and control statements for a problem can be assembled in any order but they must be end with control statement END.

Example Program 1 in CSMP

The following code shows a CSMP III program for the automobile wheel suspension problem represented by

$$M\ddot{x} + D\dot{x} + Kx = KF(t)$$

It has been coded for the case where M=2.0, F=1 and K=400. And there will be defferent runs with different value of D as specified

TITILE AUTOMOBILE SUSPENSION SYSTEM

*

PARAM D=(5.656,16.968,39.592,56.56)

*

X2DOT=(1.0/M)*(K*F-K*X-D*XDOT)

XDOT=INTGRL(0.0,X2DOT)

X=INTGRL(0.0,XDOT)

*

CONST M=2.0,F=1.0,K=400.0

TIMER DELT=0.005,FINTIM=1.5, PRDEL=0.05

PRINT X, XDOT, X2DOT

END

STOP

Example program 2 in CSMP III

Let we have a model represented by differential equation

$$\frac{dc}{dt} = \frac{C_0 - C}{\theta} - kc$$

In CSMP

Solution

TITLE SIMPLE CSTR REACTION SIMLUATION

*

PARAM ICC=0, THETA=2.0, K=1.0, C0=1.0,V=1.0

*

DCDT=(C0-C)/THETA-K*C

```
C=INTGRL(ICC,DCDT)

*

TIMER FINTIM=15.0, DELT=0.05, PRDEL=0.5

PRINT C, DCDT

END

STOP
```

Hybrid simulation

For the most studies, the model is clearly either of continuous or discrete nature; an that is determining factor in deciding whether to use an analog or digital computer for system simulation.

However, there are times when an analog and digital computer are combined to provide a hybrid simulation. The form taken by hybrid simulation depends upon the application. One computer may be simulating the system being studied while other is providing a simulation of the environment in to which the system is to operate. It is also possible that the system being simulated is an interconnection of continuous and discrete subsystem, which can best be modeled by an analog and digital computer being linked together..

The introductions of hybrid simulation require certain technological development for its use. High speed converters are needed to transform signal form one form of representation to another. As a practical matter, the availability of minicomputer has made hybrid simulation easier, by lowering cost and allowing computer to be dedicated to an application.

The term hybrid simulation is used generally of the case in which functionally distinct analog and digital computer are linked together for proposes of simulation.