Chapter 5

Simulation Languages

A computer **simulation language** describes the operation of a simulation on a computer. There are two major types of simulation: continuous and discrete event though more modern languages can handle combinations. Most languages also have a graphical interface and at least simple statistical gathering capability for the analysis of the results. An important part of discrete-event languages is the ability to generate pseudo-random numbers and variates from different probability distributions. It is important to identify the features, which are common to most of the simulation situations, especially to discrete event simulation applications. These can be identified as below:

- Generation of large streams of random numbers.
- Generation of random variates from a large number of probability distributions.
- Determining the length of simulation run and length of warming up period.
- Advancing the simulation clock.
- Scanning the event list to determine the next earliest event to occur.
- Collecting data.
- Analyzing data and setting confidence intervals.

The common features, which have to be invariably modeled in all simulations, are perhaps the cause of the development of special simulation software. The simulation software packages are designed to meet the following objectives.

1. To conveniently describe the elements, which commonly appear in simulation, such as the generation of random deviates.
2. Flexibility of changing the design configuration of the system so as to consider alternate configurations.
3. Internal timing and control mechanism, for book keeping of the vital information during the simulation run.
4. To obtain conveniently, the data and statistics on the behavior of the system.

Merits of Simulation Language

1. Since most the features to be programmed are in-built simulation language like comparatively less programming time and effort.
2. Since the simulation language consists blocks, specially constructed to simulate the common features, they provide a natural framework for simulation modeling.
3. The simulation models coded in simulation languages can easily be changed and modified.
4. The error detection and analysis is done automatically is simulation language.
5. The simulation models developed in simulation languages, especially the specific application packages, called simulators, are very easy to use.

TYPES OF SIMULATION LANGUAGE

Computer simulation essentially and experimentally used for studying a wide variety of system. The purpose of such simulation is to observe the behavior of a given system within a given environment of the system. The abstract model of the system being simulated takes the form of a computer program and the system behavior is given by the output as the program run.

We can simulate system to study their behavior using general purpose language (GPL) such as FORTRAN, C. PASCAL. But it will become difficult to program to debug and to modify when we have to simulate even a moderately complex system. So it will better to use higher lever special purpose language SPL to simulate system.

The languages design specially for simulating system offer many convenient facilities.

- They facilitate modeling
- They reduce programming
- Provide better controller
- Faster generation approach of codes.

Writing simulation in GPL might be difficult tedious and time consuming. Due to these reasons the SPL that is simulation languages are used to simulate any kind of system.There are following types of simulation languages:

1) Continuous system simulation language (CSSLS)
2) Discrete system simulation language (DSSLS)
3) Hybrid system simulation language (HSL)

1) CONTINUOUS SYSTEM SIMULATION LANGUAGE

Before digital computers come into common use, analog computers were being used for simulating continuous system. The system in and analog computers was represented by the block of electronic devices such as operational amplifier to carry out the required operation.

As soon as the digital computer arrived some of the disadvantages of analog computer are overcome using special program packages (called CANNED programs) which are implemented in digital computer to make digital computer apples like an analog computer. These are so called Digital- Analog, simulator (program package). These are meant either to replace and analog computer or check the results the of an analog simulation using Digital computer. Such simulation languages are called blocked structured continuous system simulation language.

Example: DEPI, DEPI-4, DAS, MIDAS, IBM 1130, CSMP etc.

Later more advanced expression based simulation languages were develop, example MIMIC, S/360, CSMP DYNAMO etc

2) DISCRETE SYSTEM SIMULATION LANGUAGE

These languages are used to simulate discrete system and provide the following facilities.

- Automatic generation of random no.
- Automatic data collection
- Statistical analyses of data
- Reporter generators, etc.

The other classification of discrete simulation languages is based on the general world view inherent in the language. Event, activity and process form the basis of three primary conceptual frameworks (world view) within discrete event simulation.

TYPES;

A) Event oriented language

eg. SIMSCRIPT, GASP

B) Activity oriented language

eg MILITRAN

C) Process oriented languages

eg ASPOL, SIMUFOR

D) Transaction flow oriented languages.

- Sub category of process oriented language
- System model is represented by a flow chart consisting of language.
- The program creates transaction, executes them in the blocks and makes them along the flowchart.
- These languages are flow chart oriented; the best known example is GPPS.

- **GPSS(GENERAL PURPOSE SIMULATION LANGUAGE**

-GPPS, one of the earliest DSL was developed by Geoffrey Gordon (1961-1962). The first release of this language was implemented on the IBM 704, 709 and 7090 computers. Later on improved and more powerful versions have been developed and implemented, including GPSS 2ND , GPSS 3RD (1965), GPSS 1360 (1967) and GPSS V (The latest version).

-GPSS was designed especially for those analyses who weren't necessary computer programmer because they do not write programmed in the logic as the SIMCRIPT programmer does. Instead he construct a block diagram – an n/w of inter connected blocks, each performing a special simulation oriented function.GPSS is particularly suited for traffic and queuing system.

GENERAL DESCRIPTION

GPSS Block Diagram

The development of a simulation model in GPSS is a block-by-block construction. A set of standard block is arranged in the form of a block diagram that represents the flow of entities through the various paths of the system. Each block represents a step in the action of the system and links, joining the blocks, represent the sequence of events that can occur.

To build a block diagram, it is essential to have a completed description of the system. The meanings of the blocks used in the system must be clearly defined. Each block must be assigned the block time, i.e. the time which the execution of the block will take.

In total, a set of 25 specific block types have been designed, which can be used in the construction of a block diagram. Each block type can be used any number of times in a block diagram, but the total number of blocks should not exceed 2047. On the completion of the block diagram, each block is assigned a number, between 1 and 2047, called the block number.

- The system to be simulated in GPSS is described as a block diagram in which block represents activities and lines joining the block indicate the sequence in which the activity can be executed.
- Each block performs a simulated oriented function.
- GPSS Vprovides a set of 48 different blocks, each of which can be used repeatedly.
- Each block has a name and specific task to perform.
- Each block type has a no. of data field such as A,B,C, and soon.
- Reflect the order to which they are specified.
- The entities of the system being simulated are called as transaction.
- Eg; costumer in a queuing system.
- **Typical blocks are ;**
  1) GENERATE: create transaction
  2) QUEUE: creates a queue of transaction and maintain certain queuing statistic.
  3) TABULATE: tabulates the time it took the transaction to reach that point from the time it enter.
  4) TERMINATE: removes transaction form the system.
  - There can be many transaction simultaneously block diagram.
  - A GPSS block diagram can consist of many blocks and given to each block and identification no. called a location is given to each block and movement of transaction is usually from one block is next block with next higher location.
  - The block type 'ADVANCED' is concerned with representing the expenditure of time. The program computes an interval of time called an action time for each transaction as it enters an 'ADVANCED' blocks and transaction remain at this block up to that action time.
  - The 'TRANSFER' block allows some location other than the next sequential location to be selected. The choice is normally between 2 blocks refer to as next block 'A' and 'B'.
  - Some symbols used for block types are shown below :

A GPSS PROGRAM

Example; simulation of manufacturing shop

Let us consider,in a manufacturing shop, a machine tools terms out parts at one every five minutes then as they are finished the posts go to inspector, who takes 4$\pm$3 minutes to examine each part and rejects 10% of parts. Each part will be represented on transaction and time until selected for problem will be one minute.

A GPSS block diagram for this system is as follows;

Figure Description;

A generate block is used to represent the output of machine by creating transaction every five minutes of time. The ADVANCE blocks with a mean of four and modifier of three is used to represent inspection will therefore be anyone of values 1,2,3,4,5,6,7. After completion of the inspection transaction go to or a TRANSFER with a selection factor ‘ACC’ to represented accepted parts and 10% go to another location ‘REJ’ to represented rejected parts. Since there is no further interest both location retired from the transfer blocks are terminate blocks.

GPPS coding of the above block diagram of the manufacturing shops.

MANUFACTURING SHOP

```
GERNERATE          5
ADVANCE            4,3
TRANSFER           .1, ACC, REJ
```

```
ACC   TERMINATE          1
REJ   TERMINATE          1
      START              1000
```

Fig: GPSS coding of manufacturing shop

RESOURCES IN GPSS

Facilities and storage;

A facility is defined as an entity or resources that can be engaged by a single transaction at a time. Storage is defined as an entity or resource that can be occupied by much transaction at a time up to some pre-determined limit.

Once a transaction seizes (holds) a facility any other transaction trying to seize the same facility is delayed until the first transaction releases the facility (resources). Let us consider a situation as below;

```
SEIZE          CPU
ADVANCE        7
RELEASE        CPU
```

Here CPU is a facility and it is seem thata transaction needs to use the resource for 7 times units. Any other transaction arriving at the block 'SEIZE' is refused to enter until the former transaction hasentered RELEASE block.

Resources which can beshared byseveral transactions are modeled using storage. Suppose we want to modelacomputer system which has64kb of memory then wemight declare 'MEMORY STORAGE 64' and then a request for 16kb memory might be represented by the sequences of blocks.

```
ENTER MEMORY 16
LEAVE MEMORY 16
```

As the facilities a transaction arriving at ENTER block at a time when it is used by other transactions, is delayed until the previous transaction release the necessary memory with.

A transaction controlling and facility can be interrupted or preempted by other transactions. In addition both facilities and storage can be available as occurs if the equipment they represents breaks down and can be available as occurs when, a repair has been made.

GATHERING STATISTICS;

Certain block types such as QUEUE, DEPART, MARK and TABULATE are used to gather statistic about the system performance.

When the condition for advancing a transaction is not satisfied several transactions may be kept waiting at a block and due to this the QUEUE block increase and DEPART block decreases the number in field A.

The MARK and TABULATE blocks are used to measure the length of time taken by transaction to more the system or parts of the system. The MARK block simply notes the arrival time on the transaction and the TABULATE blocks substitute the time noted by MARKED block from the time of arrival at the TABULATE block.

PROGRAM CONTROL STATEMENT;

- The first statement of GPSS i/p is a control statement with the word SIMULATE in operational field.

- When a GPSS simulation run is finished the program doesnot immediately destroyed the model. Instead it looks for more i/p following the START statement keeping the model exactly it was at completion of the run.

- I/P following the START statement can charge the model. For example,A storage capacity would be define by increasing a 'storage' statement for assigning a new value.

- The model could also be modifying by changing existing blocks.

- Certain control statement can be included between START statement will wipe out all the statistics gathered so far, but will learn the system loaded with transaction.

- Another control statement, CLEAR, will not only wine wipe out the statistics of proceeding run but will also wipe out transaction in the system so that the simulation started from the beginning at the rerun.

- The END statement is used to terminate the run.

PRIORITIES AND PARAMETERS;

PRIORITIES;

- Each transaction has one of 128 level of priority ranging 0-127. (with 0 being the lowest priority)
- At any point in the block diagram the priority can be set up using the priority block.
- The block is coded by coating the priority in field A of the priority block.

- It is also possible to design it the priority at the time transaction is created by putting the priority in the 'E' field of the generate block.
- If the field is left blank, the priority is said to zero.
- When there is competition between transaction to occupy a block the service rule is to advance transaction in order of priority and FIFO rules.

PARAMETERS;

- A transaction also has parameters. Which carry numeric data?
- Parameter can exist in four formats. They can be
  a) signed integers of full words
  b) signed integers of half words
  c) signed integers of byte size
  d) Signed floating point number.
- If no specific assignment of parameter type is made, the program creates transaction with 12 half words parameters.
- Any number of parameters of any type up to a limit of 255 can be specified by using fields 'F', 'G', 'H' and 'I'. of the generate block.
- The symbol nPx (Pxn) will called for 'n' number of parameter of type 'x', where 'x' takes the values F,H,B or L for full word, halfword byte size and floating point respectively.
- All parameter values are zero at the time a transaction is created
- The no. of parameter is given in field A of assign (ASSIGN BLOCK)
- A value is given to a parameter when a transaction enter and assign a block
- The value of parameter is take is given in field B. this value can be a specific value or can name any of the SNA's (standard numerical attributes) field C has one of the letter F, H, B, L to indicate the type of parameters.
- An ASSIGN block can add to, subtractfrom or replace the value of parameters.
- A '+' or '-'sign immediately following the parameter no. in field A indicates that the assign value is to be added or subtracted.

STANDARD NUMERICAL ATTRIBUTES (SNA)

Parameters are items of data that represent attributes of transaction. In addition, there are attributes of other entities of the system. Such as no. of transaction in storage or the length of a queue, which are made available to the program users? Collectively these attributes are called standard N.A.S.

Each type of SNA is defined by a one or two letter code and a number. For eg, the contents of storage no. 5 is denoted by s5 and the length of queue no 15 is denoted Q15

VARIABLE STATEMENT AND SNA's

In variable statement, SNA's can be combined mathematically with the operators +, -, *, ÷ and @ for addition,subtraction, multiplication, division and modular division. For eg the following statements;

- 5 VARIABLE 56+5(Q12+Q17) defines variable 5 as being the sum of current contents storage of number 6 + (plus) 5 times the sum of length of queues 12 and 17. When a floating point calculation is needed, the operation field is changed with notation FVARIABLE.
- SNA's provide the inputs to the functions, thereby allowing a great variable functional relationship to be introduced into the model.
- The values of SNA's change as the simulation process the program doesnot continually maintain current value but it computes the value of SNA's at the time they are needed.
- It is convenient to save values computed at one time or use at some later time. This can be done by the use of block type called SAVE VALUE. The block indicates in field A the number of one of many SAVE VALUE location. And in field B gives the SNA to be saved.
- The C field of save value block must carry one of the symbols XF,XH, XB or XL to indicate a full word half word bytesize or floating point type of same value, for eg the following block would save the current contents of storage no. 6 in half word SAVE VALUE no. 10,56, XH
- Some of SNA's are as follows;

L1 = The current value of clock time.

Fn = current status of facility no. n. This variable is 1 if the facility is busy and C if not.

FNn = The value of function 'n'

Kn = The integer 'n'

Ml = The transit time of a transaction

Qn = The length of queue 'n'

Sn = The current occupancy of storage ‘n’.

FUNCTIONS;

- In GPSS,there are no. of functions included each function is defined by giving two or more pairs of numbers that relate and input x to an output y
- The function can be in a continuous mode or discrete
- In a continuous mode, the program will interpolate. (Interrupt) linearly between define points allowing users to approximate a continuous function by a series first line segment.
- In a discrete mode, the function is treated as staircase function.
- If x_i and x_{i+1} are two successive points at which the function has been defined and an input value in the range $x_i<x<x_{i+1}$ will result. The value y_i being produce.

The below figure illustrate this 2 models of using function.

Continuous function Discrete function

- Any of SNA’s can be 20 a function as an input
- The choice of input is made at the time of defining the function.
- At least 2 statements are needed to define a function. The first is called FUNCTION statement and it is immediately followed by one or more function data statements. The function statement has the following format.

Field	Content
Location	Function number
Operation	FUNCTION
A	SNA to be used as input‘n’ for continuous
B	mode‘n’ for discrete mode

Where ‘n’ is the no. of points to be define

- The function data statement carry the values of x and y that define a function.

  - Values begin in column 1 with commas between x and y values and slashes between successive pairs ie x, y / x2, y2 /
  - Any no. of points may be in one statement but beyond column 71

TRANSFER MODES:

- A TRANSFER block can use value of a parameter on a function as the location to which it spends a transaction.
- To use a parameter mode, PX is put in the field A of the T.B., where x denotes the type and parameters no. is put in the field B. For function mode, function is put in field A. and function no. in field B.

GPSS TRANSFER MODES:

MODE	**FIELD A**	**FIELD B**	**FIELD C**
Conditional	BOTH	NEXT BLOCK A	NEXT BLOCK B
Parameter	Px	Parameter number	Increment
Function	Fn	Function number	increment
Unconditional		Next block A	next

TESTING CONDITION:

Sometimes it is necessary to test some condition to control the flow of transaction in the system. A block type call GATE can be used for this purpose. It can test the condition of any facility storage or logic switch.

Some of the condition that can be tested and symbols used to indicate the selected condition are;

LSN	Logic switch ‘n’ set
LRN	Logic switch ‘n Resest
UN	Facility ‘n’ in use
NU ‘n’	Facility ‘n’ in not use
SF n	Storage ‘n’ full
SNF n	Storage ‘n’ not full

- Another block type called TEST can test a variety of relationship between any two SNA's.
- The relationship's that can be testes and symbols used to represent them are shown below.

TEST blocks

G	greater than
GE	greater than equal
E	equal
NE	not equal
L	less than
LE	less than or equal

- Relationship symbol begin in column 13
- The SNA's to be related are placed in field is A and B

SIMSCRIPT PROGRAM:

SIMSCRIPT is a very widely used language for simulating discrete system. A completely new version SIMSCRIPT II was released by RAND re-operation in 1968. The latest version of SIMSCRIPT II.5.is released in 1972. This language is very FORTRAN in appearance. SIMSCRIPT II.5 can be view as a general programming language with extra features for discrete event simulation. Because of this general power and FORTRAN based, SIMSCRIPT has been widely implemented and used discrete simulation language.

SIMSCRIPT SYSTEM CONCEPTS

The system to be simulated is considered to consist of entities having attributes that interact with activities. The interaction causes event that change the state of the system. In describing the system, SIMSCRIPT uses the terms, entities attributes, sets, event routine. An entity is a program element similar to a subscripted variable. When an entity is created it can be interpreted as a genetic definition for a class of entities. Individual entities have values all attributes which define particular state of the entity attributes are named not a number. For eg we may define employee to be an entity and AGE, SALARY are attributes.

Entities can be a 2 type

1) Permanent

2) Temporary

Permanent (specify for all time the proceeds)

Temporary (specify dynamically all the program proceeds)

Temporary entities are physically created and destroyed their special statements.

Permanent entities have their attributes stored as array

Both temporary and permanent entities can belong to sets. And own sets. SIMSCRIPT uses pointer to chain together entities that are members of sets. Commands are available to manipulate these sets. The user can define sets and facilities are provided for entering and removing entities into and prompt sets.

Activities are considered as extending over time with their beginning and their end being mark as events occurring instantaneously. Each type of event is described by a event routine, each of which is given a name and programmed as a separate closed sub routine. Activities may be endogenous and exogenous.

ORGANISATION OF A SIMSCRIPT PROGRAM:

- Since event routines are closed routines some means must be provided for transferring control between them. It is done by the use of event notice.
- These event notices are created when it is determine that event is scheduled.
- An event notice exists for every endogenous events schedule to occur.
- An event notice records

i) The time the event is due to

ii) The event routine that is to execute the event

- The general manner in which simulation proceed is illustrated in below fig.

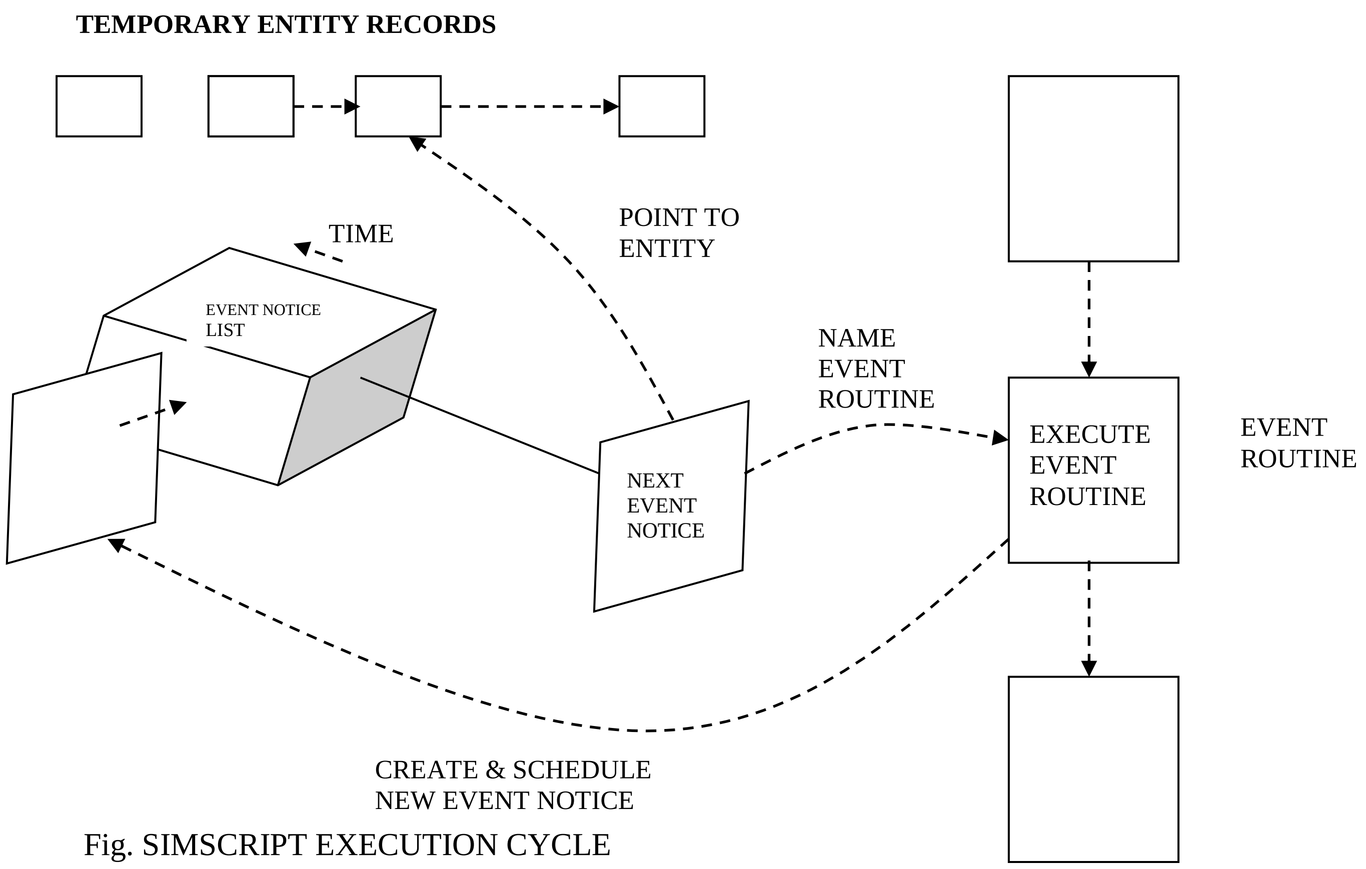

Fig. SIMSCRIPT EXECUTION CYCLE

- The event notices are filed in chronological order.
- When all events (that can be executed at a particular time.) have been proceed the clock is update to the time next event notice and the control is passed to the event routine identify by the notice.
- If the event executed by a routine results another events, the routine must create a new event notice and filed it with the other notice
- In case of exogenous events a series of exogenous statements are created. For each event this statements are similar to event notices are also filed into chronological order and are read by the program.