D.E.Ü. İ.İ.B.F. Dergisi
Cilt: 7, Sayı: 2, 1992, ss.240-255.

COMPUTER SIMULATION WITH SIMSCRIPT II.5

Vahap TECİM (*) Agata MARTIN (**)

Abstract

Nowadays it is necessary to use computers for simulation application. The purpose of this article is to list the most common simulation languages and to give more details of a useful discrete simulation language called SIMSCRIPT II.5 which has two main objects: an active object, which is called PROCESS (e.g. customer in a supermarket) and a passive object, which is called RESOURCE (e.g. clerks or cashiers in a supermarket). The main problem is to understand and manage the behaviour of these objects and their interactions. Simscript II.5 can solve complicated discrete simulation problems but in this article we would like to give only the essential commands. One can solve the same problem in different ways with Simscript II.5

A . INTRODUCTION

Simulation is the process of imitating the behavior of the real system by contructing and experimenting with a model that is only a simplified representation of the system.

Anybody who can do the modelling of the system (i.e. who knows how it evolves) can built a computer simulation. This person (or group of persons) is an expert who knows how the elements of the system interacts together.

From simulating the system, we may want to find a solution to a problem by trying out different possible alternatives. Also,we do not want to experiment on the real system; that can be very expensive or even dangerous in some cases and sometimes, the system may not even exist. The main goal of computer simulation is to find a satisfactory solution that can be implemented.

Computer simulation can either be discrete (state changes in the physical system are represented by a series of discrete changes) or continuous (system changes happens depending on the time). For example, some discrete simulation systems would be an airport, a bank, a supermarket and some continous simulation systems would be more like an automobile suspension or the ejection of a pilot from an airplane. In the case of continuous simulation,the theoretical

(*) Arş. Gör. D.E.Ü. İ.İ.B.F. Ekonometri Bölümü
(**) Department of Science University of Ottowa

equations has to be known and may include derivatives.

When somebody wants to simulate a system, he (she) should follow these steps:

a) System analysis,
- Goals
- Measures of effectiveness
- Interrelationships, constraints, and experimental frame
- Solution strategy

b) Formulation of model,

c) Model verification, validation, and certification,

d) Implementation and documentation.

The most common computer simulation languages for:

Continuous System Simulation Languages are:
- Continuous Systems Simulation Language (CSSL, .CSSL IV)
- Continous Systems Method of Programming (CSMP)
- DYNAMO
- ACSL

Discrete System Simulation Languages are:
- SIMSCRIPT (I.5, II, II.5)
- Generalized Approach to Simulation Programming (GASP,II, III)
- Computer Simulation Language (CSL)
- General-Purpuse Simulation Language (GPSS)
- SIMULA
- SLAM, SLAM II
- SIMAN

General programming languages like FORTRAN, PL/1, PASCAL, C and BASIC can also be used for simulation purposes.

We will now discuss SIMSCRIPT II.5 which is one of the most common discrete high level simulation languages.

B. SIMSCRIPT II.5

SIMSCRIPT II.5 is a discrete system simulation language. It has been developed in 1960 by CACI Inc. and can be used on both mainframe and personel computers. The latest and the mosι comprehensive version is Simscript II.5 which is divided into five levels: the first three levels provide general-purpose programming capability; the fourth level contains powerful list processing facilities for entity manipulations, and the fifth level is oriented towards discrete system simulation.

1.General Characteristics of Simscript II.5

Simscript II.5 can be considered as a general-purpose language (like Fortran, Pascal, C or PL/1) to which was added some simulation programming features.

General features of Simscript II.5: English-like, free-form, modular (conceptual and separete compilation) and dynamic storage.

Every statement begins with a keyword.

Ex: **ACTIVATE A CUSTOMER NOW**

START SIMULATION

There are two kinds of variables that can be used with Simscript, system variables and user defined variables:

a) System Variables: All end in "." "letter"

PI.C	(Constant)
HOURS.V	(Variable)
SORT.F	(Function)
TIME.R	(Routine)
EV.S	(Set)

b) User Defined Variables: Any content of letters, digits and periods can have up to 8 characters. For example, COSTUMER, NO. OF. CUSTOMERS, V34, V23.5. A simulation typically includes variabιes of two classes.

- **Time Variables** changes discontinuously with time over the course of the simulation. Ex. number of customer in a queue (ie. queue length), the utilization of a resource (bank teller, CPU).

- **Sample Variables** maintains the collection of measurements of some attribute which occurs during the simulation, of the entities within a particular class. Ex. the waiting time of people in a queue, the time needed for a computer to execute a job.

2. The Simscript Program Structure

PREAMPLE

Define all model components statically (processes, resources, entities, events and sets)

Define all global variables

Define all statistics to be gathered

END

MAIN

Execution starts here

:

START SIMULATION (now simulation time starts)

:

(Pass control to timing routine. Here can be writen all outputs - results of the simulation)

:

END

(Timing routine)

PROCESS: The description of an object and the sequence of activities which it experiences throughout its "life" in a simulation. A process models in object (static) and its "life" in the system (dynamic). A simulation may contain many copies of a single process and/or many different processes. Processes interact by: Changing system state, direct communication and compiling for resources. Here are some simple process commands where NEW.CUSTOMER is the process name.

ACTIVATE A NEW.CUSTOMER NOW

ACTIVATE A NEW.CUSTOMER IN 5. HOURS

RESOURCES: The description of an object which provides services to the process objects. Acting to delay them if already occupied serving another process object. A simulation may contain many different resources and/or many units of each resource serving a single queue of processes. Here are some examples showing you how to use the resources:

REQUEST 1 UNIT OF PUMP (1) or REQUEST 1 PUMP (1)

REQUEST 5 WORKERS

RELINQUISH 1 TELLER

ENTITIES: Passive objects moved through the system (by process).

EVENTS: Instantaneous processes.

SETS: Ordered lists of entities, processes or resources. Sets the default setting is a First In First Out (FIFO) queue but it can be changed for Last In First Out (LIFO) queue.

4. Some Important Simscript II.5 Commands

4.1 ACCUMULATE and TALLY statements:

The statements ACCUMULATE and TALLY provide the means for conveniently acquiring various measurements where the variable can be either a time variable (for the Accumulate statement) or a sample variable (for the tally statement).

Syntax: {ACCUMULATE /TALLY] storage. variable AS THE {Number, Sum, Mean, Maximum, Minımum} OF variable.

4.2 CREATE and DESTROY statements:

The CREATE statement is used to create a process notice for a particular type of process without placing it into the event list.

Ex: CREATE A CUSTOMER

The CREATE statement is used to create entities. This command is to be used in order to create resources.

Ex: In order to create 3 PUMP resources we can use one of those two block commands:

a) CREATE EVERY PUMP (3)

b) LET N. PUMP=3

CREATE EVERY PUMP

c) PREAMPLE

RESOURCE INCLUDE PUMP EVERY PUMP HAS A GRADE AND HAS A RESERVE (Where grade and reserve are pump's attributes. Note that resources does not necessarily have such attributes.)

END

MAIN

CREATE EVERY PUMP (3)

END

Simscript will be creating a table for the above sequence of commands as the one shown below

(1)	(2)	(3)	
			U.PUMP
			N.X.PUMP
			N.Q.PUMP
			GRADE
			RESERVE

Where:

U.PUMP(I): Gives the number of "copies" of the ith variety of PUMP that exists. Is assigned by the user writing commands such as LET U.PUMP (1)=6, LET U.PUMP(2)=2 and LET U.PUMP(3)=4.

N.X.PUMP(I): Maintains the current value of the number of the variety of PUMP that are busy accommodating a process that requested the resource.

N.Q.PUMP(I): Maintains the value of the number of requests for the ith variety of PUMP that are currently waiting for the resource.

Note that for all , U.PUMP(I)=N.X.PUMP(I)+N.Q.PUMP(I) at all time during the simulation.

The DESTROY statement is used to destroy (i.e. to make non existent) a process notice which is in the "created" state.

Ex: DESTROY THE CUSTOMER

4.3 ASSIGNMENT (LET):

Relational operators IF, FOR, DO-LOOP statements are the same as Fortran.

Ex:
```
FOR EACH PUMP
    DO
    LET RESERVE (PUMP)=1000
    LET U.PUMP(PUMP)=1
    LOOP
```

4.4 LOGICAL TESTS AND SEARCH STRUCTURE:

Ex 1:
```
IF GRADE (PUMP)=REGULAR AND RESERVE(PUMP)<5600
    OR N.Q. PUMP(PUMP)>0
        :
    ALWAYS
```

Ex 2:
```
FOR EACH PUMP,
    WITH GRADE (PUMP)=REGULAR AND RESERVE(PUMP)>0
    FIND THE FIRST CASE
    IF FOUND
        :
    ELSE
        :
    ALWAYS
```

Ex 3: If we wish to know which PUMP has the shortest queue, we must write,

```
FOR EACH PUMP,
COMPUTE MY.CHOICE AS THE MINIMUM(PUMP) OF
N.Q.PUMP(PUMP)

      :

REQUEST 1UNIT OF PUMP(MY.CHOICE)
```

4.5 ACTIVATE statement:

This statement is used to create a process notice for a particular process and the immediately place the notice in the event list.

Ex: ACTIVATE A CUSTOMER IN 25.MINUTES

ACTIVATE A CUSTOMER IN EXPONENTIAL. F (2 . 5 , 1). MINUTES

4.6 REQUEST and RELINQUISH statement

The REQUEST statement is used by a particular process entity in its process routine to request a certain number of units of a specified resource. If the number of resources requested are avaliable, then these units are activated and the process entity immediately moves to the next statement in its process routine (usually a work statement). If the requested units are not all available, then the process entity is put into the queue for the requested type of resource and control is returned to the timing routine.

Ex: REQUEST 1 UNIT OF CASHIER(1)

REQUEST 1 CASHIER(1)

The RELINQUISH statement is used by a particular process entity to relinquish a certain number of units of a resource of a specified type. If a queue corresponding to the relinquished type of resource is not empty, the resources made available by the relinquish statement can be used by another process which had requested the resources that were unavailable.

Ex: RELINQUISH 1 UNIT OF CASHIER(1)

4.7 WORK and WAIT statements:

Those statements are used by a particular process, which is executing and it prevents itself from doing anything else for a specified amount of time. A WORK statement represents **service** times and a WAIT statement represents **interevent (interarrival)** times.

Ex: WORK 40.MINUTES or WAIT 2.HOURS

4.8 SUSPEND and REACTIVE THE statements:

The SUSPEND statement is used by a particular process to stop its own execution for an unspecified amount of time.

Syntax: SUSPEND

The REACTIVATE THE (or THUS) statement is used in a process or normal routine to reactivate previously suspended process(es).

Ex: REACTIVATE THE CUSTOMER NOW

4.9 INTERRUPT and RESUME statements:

The INTERRUPT statement is used in a process or normal routine to delay the activation of a process other than the one where the statement is called from.

Ex: INTERRUPT THE CUSTOMER

The RESUME statement is used in a process or normal routine to resume an interrupted process.

Ex: RESUME THE CUSTOMER

4.10 END and RETURN statements:

The END statement is placed at the end of each process routine.

Syntax: END

The RETURN statement may be placed in the **middle** of a process routine, if executed by a process entity, it has the same effects as the END statement.

Syntax: RETURN

4.11 DEFINE and DEFINE TO MEAN statements:

The DEFINE statement define every variables which are either integer, real or characters. The variables can be either global or local. When a variable is defined in the PREAMPLE it is a global variable but when it is defined in any subprograms other than PREAMPLE it is local to that subprogram.

Syntax: DEFINE var1 (var2,...) AS [A] {REAL,INTEGER,TEXT}

The DEFINE TO MEAN statement can be used to substitute a character string for a single word, thereby making the program more readable.

Ex: DEFINE IN. COMING TO MEAN 1

We can also define the time units for the simulation clock which is given by a variable called TIME.V. The default unit of time for TIME.V is day, which means that any output variables(e.g. mean delay of customer in queue) will also be in days. User can change the mean units by using:

Ex: DEFINE .MINUTES TO MEAN UNITS

In Simscript, we can use 3 default simulation time units: DAYS, HOURS and MINUTES. Simscript knows that 1 Day=24 Hours and 1 Hour=60 Minutes. The system variable TIME.V is the system's clock and the flexibility of the system allows the user to redefine the time units.

Ex: DEFINE .SECONDS TO MEAN HOURS.

4.12 RANDOM.F and SEED.V statements:

The RANDOM.F statement is a random number generator in SIMSCRIPT. It generates numbers between 0 and 1.

Ex: LET V=RANDOM.F(I)

where I, an integer between 1 and 10, is the number of the desired stream. Random number stream I is initialized by the SEED.V(I) (for I=1,2,...,10) with each new submittal of a situmulation model. It is very important for any stochastic simulation model to have a good random-number generator, that is a mechanism for generating independent samples from a uniform distribution on the interval [0,1].

Ex: LET T1=SEED.V(1)

4.13 Status of Process

During the simulation, the user can control all the processes. Using the built-in command STA.A, one can obtain the actual status of a process:

STA.A(process)=0 when the process is **not active** (waiting stage)

=1 when the process is **active** (working stage)

=2 when the process is **suspended**

=3 when the process is **interrupted**

4.14 Statistical Information

Simscript II.5 can use standard discrete distribution (Table 4.14.1) and/ or standard continuous distribution (Table 4.14.2) defined below and where I is a stream number.

Table 4.14.1 Standard Discrete Distribution in Simscript

Distribution	Range	Function Name and Argument
Discrete Uniform	A,A+1,...,B	RANDI.F(A,B,I)
Poisson	0,1,...	POISSON.F(MEAN,I)
Binominal	0,1,...,N	BINOMINAL.F(N,P,I)

Table 4.14.2 Standard Discrete Distribution in Simscript

Distribution	Range	Function Name and Arguments
Exponental	[0,∞)	EXPONENTIAL.F(MEAN,I)
Gamma	[0,∞)	GAMMA.F(MEAN,ALPHA,I)
Erlang	[0,∞)	ERLANG.F(MEAN,ALPHA,I)
Weibull	[0,∞)	WEIBULL.F(ALPHA,BETA,I)
Lognormal	[0,∞)	LOG.NORMAL.F(MU,SIGMA,I)
Normal	(-∞,∞)	NORMAL.F(MU,SIGMA,I)
Beta	[0,1]	BETA.F(ALPHA1,ALPHA2,I)
Uniform	[A,B]	UNIFORM.F(A,B,I)

```
Ex1: Do I=1 to 500
        ACTIVATE a customer now
        WAIT EXPONENTIAL.F(1,1) .minutes
        Loop

Ex2: REQUEST 1 pump
        WORK UNIFORM.F(1,4.7) .hours
        RELINQUISH 1 pump
```

5.Comments

In conclution, Simscript II.5 is very useful for simulating a system and the commands are very easy to use and very flexible. But before deciding to use SIMSCRIPT II.5 you should take into consideration the following:

1) The mainframe version does not have any graphic interface but is fast in giving the results.

2) The PC version has the graphic interface but it is quite slow even with small problemes.

3) The available documentation is hard to understand and incomplete.

4) The "debugging" of a Simscript program can be very time consuming because the error codes do not give the user the knowledge of the exact cause of the problem.

ÖZET

Günümüzde simulasyon uygulamalarının bilgisayarlarla yapılması zorunluluk haline gelmiştir. Bu makalenin amacı genel kabul görmüş simulasyon programlama dillerini listelemek ve kullanımı oldukça yaygın olan kesikli programlama dili SIMSCRIPT II.5 hakkında daha detaylı bilgi vermektir. Simscript II.5 iki ana objeye sahiptir: Aktif obje, PROCESS olarak adlandırılmakta (süpermarketteki müşteri) ve pasif obje RESOURCE olarak adlandırılmaktadır (süpermarketteki kasiyer). Esas problem bu iki objenin

hareketlerini anlayıp yönetebilmek ve birbirleri arasındaki iletişimi sağlayabilmektir. Simscript II.5 karmaşık kesikli simulasyon problemlerinin çözme imkanına da sahip olmakla birlikte biz bu makalede sadece önemli sayılabilecek komutları vermek istiyoruz. Ele alınan problem Simscript II.5 ile değişik yollardan çözülme imkanına sahiptir.

C.REFERENCES

BRAUN, Jay E.; **Simscript II.5 Reference Handbook.**C.A.C.I.Inc., Los Angeles ,1983.

LAW, A.M. and KELTON, W.D.; **Simulation Modelling and Analysis.** McGraw-Hill, New York ,1987.

LAW, A.M. and LARNEY, C.S.; **An Introduction to Simulation using SIMSCRIPT II.5,** C.A.C.I. Inc.,La Jolla, CA 92037, 1985.

Mac DOUGALL, M.H. **Simulation Computers Systems: Techniques and Tools.** MIT press, Cambridge 1987.

MULLARNEY, Alasdar and JOHNSON, Glen D.; **Simscript II.5 User's Manual, IBM S/ 370 series.** C.A.C.I. Inc.,Los Angeles, 1984.

NEELAMKAVIL, Francis; **Computer Simulation and Modelling.** John Willey, Chichester. 1988.

O'REILLY, Jean J.; **Slam II, Quick Reference Manual.** Pritsker & Associates Inc. West Lafayette, 1985.

PRITSKER, A.A.B.; **Introduction to Simulation and SLAM II.** John Wiley, New York, 1988.