INTRODUCTION TO SIMULATION LANGUAGES

Robert G.Sargent

INTRODUCTION

The purpose of this paper is to give a brief introduction to simulation languages. The paper contains a discussion on the hierarchy of computer languages and their relation to simulation, the advantages and disadvantages of using simulation languages, factors to consider when selecting a language, and some of the characteristics of the three most popular discrete simulation languages. A more thorough treatment of simulation languages and their use are contained in [2], [4], and [10].

A significant portion of the effort in any simulation study is in programming the model to run on a digital computer. This programming occurs after the model has been defined. The selection of the language should, in general, be done prior to developing the model to be programmed because (i) there must be a comptibility between how the system is modeled and the computer language used and (ii) some simulation languages aid in the modelling process. The language selected is usually a general purpose language (higher level language) or a simulation language.

HIERARCHY OF COMPUTER LANGUAGES

The digital computer operates in a language called machine language which consists of 0's and 1's. We do not program in this language due to its complexity. The next level of computer languages is assembly languages and they are usually machine dependent. Programs are written in assembly language only if they are to be used over and over again because the programming effort required is considerable. Analysts almost never used assembly language to program simulation models.

The next level of computer languages is general purpose or higher level languages. They are user oriented and are usually machine independent. Machine independent means that if an application program is written in one of these languages, that program will generally run on any computer having that language. General purpose languages are either compiler or interpreter languages. A compiler language uses a compiler to convert application programs into machine language or into assembly language for conversion to machine language. This compiling (conversion) takes a certain amount of computer time. The computer then "executes" the resulting machine program.

In interpreter languages, each line of the application program is converted to machine language each time it is executed. This means that if a line of an application program is used several times, it must be converted to machine language each time. Interpretive languages generally require more computer time for discrete event simulation than compiler languages because a considerable portion of discrete event simulation programs are used over and over again. Figure 1 contains examples of compiler and interpretive general purpose languages.

General purpose languages are frequently used in programming simulation models with Fortran being the most commonly used one for discrete event simulation. In fact, Fortran may be the most used of all languages for discrete event simulation. General purpose languages are also the implementation language of several of the simulation languages, e.g., GASP.

The next level of computer languages are problem or application oriented in addition to being user oriented. Simulation languages belong to this level of languages. Beginning in the late 1950's and earlier 1960's, different groups of individuals performing simulations recognized that several of the same functions were used in almost every simulation and they therefore could be programmed into subroutines and "tied" together to be used for future simulations. From these special programs (languages, if you wish), simulation languages have evolved until today we have commercially available several general and special purpose simulation languages, including some extremely sophisticated ones. Some examples are given in Figure 1.

General purpose simulation languages can be broken into three general classes: discrete, continuous, and combined (discrete/continuous). Discrete simulation languages are for programming discrete event simulation models, i.e., simulation models whose "states" change at specific points in time. Continuous general purpose simulation languages are for models whose variables (states) change continuously over time. Combined languages are of recent development and provide the capability of allowing models to have some variables that change continuously over time and others that change at specific points in time.

Special purpose simulation languages are simulation languages that have been developed for modelling

Simulation Languages (continued)

FIGURE 1

Hierarchy of Computer Languages With Examples

- Machine Languages
- Assembly Languages
- General Purpose Languages
  - Compiler:
    - Fortran, PL/1, BASIC, ALGOL, COBOL
  - Interpreter:
    - APL
- Simulation Languages
  - General Purpose
    - Discrete:
      - GPSS, GASP, SIMSCRIPT, SIMULA
    - Continuous:
      - DYNAMO, CSMP, MIMIC, MIDAS
    - Combined (Discrete-Continuous):
      - GASP IV, C-SIMSCRIPT
  - Special Purpose:
    - CSS II, ECSS, BOSS, Q-GERT

and simulating specific systems. Specialized simulation languages for simulating computer systems are examples of these languages. Many of the special purpose simulation languages have evolved from general purpose simulation languages.

The remainder of this paper, unless otherwise stated, will be restricted to the use of general purpose and discrete simulation languages for discrete event simulations.

WHY SIMULATION LANGUAGES

The major advantage of using simulation languages over other languages is the reduction in programming time required to program the model. This is extremely important as it allows the analyst performing the simulation study to devote more time to other phases of the study. Some of the simulation languages provide, in addition, conceptual guidance, modelling capability, and aid in communication and documentation.

The major disadvantages of using simulation languages are (i) that analysts must learn the simulation languages they plan to use, and (ii) the cost of obtaining them. The computer time for using a simulation language may be more and rarely is less than using a general purpose language.

Simulation languages generally provide at least the following functions:

(a) Generation of Random Numbers
(b) Generation of Random Variates
(c) Time Flow Mechanism (Advance time and keep a list of future events)
(d) Collect Data for Analysis
(e) Perform Analysis on Collected Data
(f) Provide Error Diagnostics

SELECTION OF A LANGUAGE

There are two different levels in selecting languages for simulation. The first level is concerned with what languages should be available for simulation in a given organization. The second level is what language should an analyst use in a specific simulation study.

Some of the factors that need to be considered in selecting languages for an organization are:

(1) Language compatibility with organization's computer system;
(2) Language adequately supported;
(3) Language suitable for problems that likely will be simulated;
(4) What are the costs to obtain, install, maintain, and update the language;
(5) Difficulty in learning language;
(6) Documentation on language;
(7) Language computer time efficiency;
(8) Language flexibility;
(9) Language capability, including error diagnostics, modelling capability, data analysis, etc.;
(10) Will use of the language justify its cost.

In selecting a language for a specific problem the analyst generally considers at least the following:

(1) What is available, either inhouse or commercially available elsewhere such as on time sharing systems.
(2) What languages does the analyst know.
(3) What type of problem does the analyst have.
(4) What are the language capabilities including:
    (a) which world view: event, activity, or process
    (b) problem compatibility
    (c) data collection and data analysis
    (d) ability to expand model, if necessary
    (e) generation of random numbers and variates
    (f) error diagnostics and documentation
    (g) communication ability, particular to user of model's results.
(5) Programming effort required.
(6) Computer time required.

The analyst is generally able to quickly reduce the choice of a language down to at <u>most</u> one special purpose simulation language, one general purpose simulation language, and one general purpose language. The analyst then will usually choose the language requiring the least programming effort provided it will allow flexibility to expand the model in the future, if it should ever need it, and the computer time required to use that language is reasonable.

SOME SPECIFIC SIMULATION LANGUAGES

The most popular discrete simulation languages in the United States today are GASP, GPSS, and SIMSCRIPT. These languages were born in the early 1960's and they continue to evolve. Each have their own strengths and weaknesses. Table 1 contains some of the characteristics of the current versions of these languages.

TABLE 1

Characteristics of Some Simulation Languages

Characteristic	GASP	GPSS	SIMSCRIPT
Current Versions	GASP II (Discrete), GASP IV & GASP-PL/1 (Discrete, Continuous & Combined)	GPSS/360, GPSS V, GPSS-H, and several others (All Discrete)	Simscript II.5 (Discrete), C-Simscript (Discrete, Continuous & Combined)
Implementation Language	Fortran or PL/1	Assembly	Assembly
Computer Usage	Any computer having Fortran or PL/1 compilers	Most large computers (some commercial time sharing services)	Most large computers (some commercial time sharing services)
Language Orientation	Statement	Block	Statement
Discrete World View	Event	Process (Transactions)	Event or Process
Storage Management	Fixed	Dynamic	Dynamic
Language Type	Compiler	Interpreter	Compiler
Language Cost	Inexpensive	Moderate	Moderate
References	[5, 6, 7]	[1, 3, 9]	[8]

CONCLUSIONS

Simulation languages are used in a vast majority of simulations and their use is increasing. Simulations languages have always evolved and this trend will continue in the foreseeable future. The recent availability of combined (discrete/continuous) simulation capability in GASP was extremely well received by analysts. Currently, several of the simulation languages are being developed for use on minicomputers. The use of computer graphics and interactive capability will probably become popular in the future.

BIBLIOGRAPHY

1. Bobllier, P.A., B.C. Kahan, and A.R. Probst, Simulations with GPSS and GPSS V, Prentice-Hall, 1976.
2. Emshoff, J.R. and R.L. Sisson, Design and Use of Computer Simulation Models, McMillian, 1970.
3. Gordon, G., The Application of GPSS V to Discrete System Simulation, Prentice-Hall, 1975.
4. Gordon, G., System Simulation, 2nd Edition, Prentice-Hall, 1977.
5. Pritsker, A.A.B., The GASP IV Simulation Language, John Wiley & Sons, 1974.
6. Pritsker, A.A.B. and P.J. Kiviat, Simulation with GASP II, Prentice-Hall, 1969.
7. Pritsker, A.A.B. and R.E. Young, Simulation with GASP-PL/1, John Wiley & Sons, 1975.
8. Russell, E.C., Editor, Simscript II.5 Programming Language, CACI, 1973.
9. Schriber, T.J., Simulation Using GPSS, John Wiley & Sons, 1974.
10. Shannon, R.E., System Simulation: The Art and Science, Prentice-Hall, 1975.