CHAPTER 1.

SIMULATION AND DISCRETE EVENT SYSTEMS.

1.1 The SIMULA Project.

The two main objects of the SIMULA language are :

1. to provide a language for a precise and standardized description of a wide class of phenomena, belonging to what we may call "discrete event systems".

2. to provide a programming language for an easy generation of simulation programs for "discrete event systems".

SIMULA is based on ALGOL and contains that language as a subset. The extension consists of the introduction of some new basic concepts, new statements which operate upon these concepts and a set of library procedures. In order to achieve the greatest possible logical integration of SIMULA and ALGOL, the scope of some of the ALGOL statements has been extended to operate also upon the new concepts introduced.

SIMULA has been developed by the authors at the Norwegian Computing Center, since the summer of 1963 under a contract with UNIVAC Division of Sperry Rand Corporation. A SIMULA compiler for the UNIVAC 1107 and 1108 Computers is completed and has been in operation since December 1964.

In addition to the authors Björn Myhrhaug at the NCC, Bernard Hausner (now at RAND Corporation) and Ken Walter (now at Case Institute of Technology) have taken part in various stages of the project.

The authors also want to express their gratitude to all the programmers and operational research workers who have taken interest in the project and contributed useful ideas and comments.

1.2 Simulation.

Before the appearance of the electronic computer, properties of telephone communication systems, ticket counter systems, machine systems, etc. were mainly analyzed by analytic means.

In mathematics the theory of stochastic processes furnished the basic tools for work in these areas. Important and interesting results were, and still are obtained by this approach.

The technological development and the introduction of scientific methods for planning and decision making has created a need for study of complex systems. The usefulness of the analytical approach is, however, limited to rather simple situations. The electronic computer has made simulation an extremely powerful method of analysis, and the use of simulation has been extended to many other areas than those mentioned above by new disciplines like operational research. By simulation it is possible to study very complex problems, and both transient and stationary states of systems may be analyzed whereas the analytical approach often is limited to stationary states.

This does not imply that analytical studies have become obsolete. If a solution is obtainable from a realistic analytical model, it offers more complete and reliable information than the statistical inference from a simulated sample of system runs.

Since it is possible to simulate very complex situations, it is tempting to substitute a "too realistic" simulation for clear thinking and valid simplification. This is done by the introduction of so many variables and variations in decision

rules that the basic structure of the problem is obscured and no certain information on the important features of the system is obtained.

Also, one is tempted to forget that in many situations (e.g. in military operational gaming) the number of available combinations of strategies is so large that only a very small percentage may be simulated.

The programming of a simulation may be very time-consuming. This is not too serious if a large number of alternatives are to be run and the model of the actual system and the decision rules to be used are well defined.

This is as a rule not the case. On the contrary, one wants to experiment with different layouts and decision rules, trying to understand the system, gradually introducing more complexity in those parts of the system where this is essential. Very often it is found that apparently minor changes in the system call for extensive reprogramming, and the usefulness of the approach is greatly reduced.

The advent of the algorithmic languages like ALGOL and FORTRAN has only slightly improved the situation. These languages are basically operating on fixed data structures and have simpler principles for sequencing of actions than needed in simulation.

General simulation programs have been developed. Some are rather ad hoc tools for faster generation of programs, other try to create simulation languages. The best known languages are GSP by Tocher and his colleagues, CSL by Buxton and Laski and SIMSCRIPT by Markowitz, Hausner and Karr. The authors have gained from their aquaintance with these efforts.

SIMPAC by Lackner was not available to the authors. GPSS by Gordon is so different from SIMULA in its approach that it has not had much influence on the latter.

The last, but perhaps the most important restriction in the use of simulation is the lack of a basic language for problem formulation. Programming cannot be started before the system is precisely described. There is a strong demand for basic concepts useful in understanding and describing all systems studied by simulation from a common point of view, for standardized notation, for easy communication between research workers in different disciplines.

SIMULA is an attempt to meet this demand. Problem formulation and not program generation has been our starting point. This being said, it is clear that the language should be designed so that system descriptions may produce simulation programs through a compiler.

1.3 SIMULA Design Objectives.

1. Since simulation is the method of analysis most commonly to be used for the systems in which we are interested, SIMULA is a dynamic language:

   It is designed to describe sequences of actions, not permanent relationships. The range of variation in decision rules and interactions between components of systems is so wide that it is necessary to let the language contain a general algorithmic language. An important reason why ALGOL has been chosen is that its block structure is similar to what was needed in SIMULA.

2. SIMULA should be built around a few basic concepts, selected to provide the research worker with a standardized approach to a very wide class of problems and to make it easy to identify the various components of the system.

3. Attempts have been made to make the language unifying - pointing out similarities and differences between systems, and directing - forcing the research worker to consider all relevant aspects of the systems. Efforts have also been made to make SIMULA descriptions easy to read and print and hence a useful tool for communication.

4. Taking the above objectives into account, SIMULA descriptions (supplemented by the necessary input, output and data analysis statements) should without any rewriting be able to produce simulation programs for elctronic computers through compilers.

1.4 Discrete Event Systems.

In our discussion of language concepts we will start with a well-known system:

An office with a series of ticket counters, offering a range of services to customers. Customers arrive in the system with time intervals described by an arrival distribution, they enter a queue and they are given service by a clerk behind a counter after some time in the queue. Then the customer disappears from the system or enters another queue.

The customers are moving through the system, the clerks remain. The queues also remain but their contents change.

One may regard the clerks as the active partners in the interaction taking place, pushing the passive customers from place to place.

The same point of view also is a natural one when studying a stream of materials through a factory. The materials are passive, the machines are active.

The customers (or the materials) may all be exactly similar, but often they must be characterized by priorities, by a description of the service they demand, their earlier history in the system etc. They are carriers of information.

This distinction between passive entities being carriers of information and active entities acting upon and pushing the passive ones around is so natural in many systems commonly studied that some simulation languages have introduced passive data carriers and active "machines", "stations" or "routines" as two basic concepts. This was also done in the early stages of SIMULA.

In social systems like epidemics, attitude diffusion etc., we encounter entities ("individuals") interacting with other entities of the same kind and at the same time acted upon by other entities (e.g. "treatments").

When we extend the counter system or the factory, it may be natural to regard the entities passive in one part of the system, as active in another part or in another situation. Also, it is always possible to reverse the point of view and describe the customers and the materials as the active partners, acquiring service from the passive clerks or machines only characterized by data describing their operational properties.

For these reasons, and to achieve greater flexibility and unity, SIMULA has integrated the two kinds of entities into one. The basic concept in SIMULA is the process, being characterized by a data structure and an operation rule.

The individual items of the data structure of a process will be called attributes.

A process may be active in some stages of its presence in the system, passive in others, depending upon its operation rule and interference with other processes. It is possible to let the operation rule of a process be rudimentary and in this way obtain an always passive data carrier as described above.

During an active phase of a process it may "connect" other processes, making their attributes available to itself for decision-making and manipulation. During an active phase a process also may "schedule" a later active phase for itself and other processes.

An active phase of a process is called an "event". The scheduling of an event is made by an "event notice" telling at which "system time" the event will occur and to which process it refers.

In SIMULA the system time is kept constant during an event, and hence all actions during an active phase may be regarded as instantaneous.

Thus SIMULA may be used to describe systems which satisfy the following requirement:

<u>The system is such that it is possible to regard its operation as consisting of a sequence of instantaneous events, each event being an active phase of a process</u>.

The number of processes may be constant or variable and they all belong to one or more classes called activities.

Since the set of system times at which events occur forms a discrete point set on the system time axis, and since every action in the system is a part of an event, we will name these systems discrete event systems.

By introducing suitable processes SIMULA also may be used to describe with the desired degree of accuracy continuously changing systems, as well as systems in which some processes have continuous changes, other processes discrete changes.

In addition to processes we need entities which may be used as "storing places" for processes: queues, files etc. The SIMULA "set" concept serves this purpose. The contents of a SIMULA set may consist of any mix of different kinds of processes and may be changed during the operation of the system.

We also want to be able to assign names to specific processes and to refer to processes through their event notices.

Because of the need for uniformity in the logical structure and the implementation on a computer, all references to processes in SIMULA are indirect, through a standardized reference called an "element".

To obtain a suitable system description, we may often need more than one element referring to the same process: The clerk at an airport check-in counter has to check if the passenger he is giving service also is referred to in the list of booked-in passengers. We may even want to have more references to the same process in a given set: In the police criminal records a person may be present under different aliases.

The naming of specific processes in SIMULA is achieved through "element variables" taking on specific elements as values.

Having introduced processes, event notices, sets, elements and element variables as basic concepts, we need statements utilizing and operating upon these concepts to be able to give a precise description of the data structures and operation rules of processes. Part II of this report gives precise definitions of the basic concepts and available statements in SIMULA.