DEVELOPMENT OF THE GPSS-ENVIRONMENT
THROUGH THE OBJECT ORIENTED APPROACHES OF C++

1 INTRODUCTION

1.1 SIMULATION

A visionary man like Hamming [17] has described the potential tools and working habits that are only emerging now - almost forty years ago:

"In time, especially in fields where the basic theory is well understood, at least nine out of ten experiments will be done on computers, and only the interesting, suggestive experiments will be done in the laboratory to check the theoretical model... . In the long run this change from the physical approach to the conceptual approach will revolutionize much of science and engineering and will keep the computers busy performing the clean-cut, idealized laboratory experiments whose interrelationships are too complex for the human mind to work its way through unaided, as well as the complex, messy situations that occur in engineering practice"

The above statement emphasises the importance of simulation in the research environment.

Simulation is the technique of constructing and running a model of a real system in order to study the behaviour of
that system, without disrupting the environment of the real system. Simulation is experimentation with models, typically models set up on a digital computer [27].

Simulation encompasses a model building process as well as the design and implementation of an appropriate experiment involving that model. Thus simulation permits inferences to be drawn about the system being modelled [37].

The verb "to simulate" is used to describe the ancient art of model building. Naylor [32] defines simulation as a numerical technique for conducting experiments on a digital computer, which involves certain type of mathematical and logical models that describe the behaviour of a business or economic system (or some component thereof) over an extended period of time. For such situations where the major characteristics of the problem being faced are complexity, poor definition and mathematical unpleasantness, simulation is a technique which can be gainfully employed [53]. Both analytical and simulation models have been developed for modelling systems like flow line production systems. As the problems becomes more and more complex, the analytical approach becomes intractable [45].

Payne [36] comments that though computer methodology is only a part of the work required in simulations, it has given rise to some of the largest and most logically complex
computer programs ever developed and thus, simulation is an important area of computer science.

The principal difference between a simulation experiment and a "real world" experiment is that, with simulation, the experiment is conducted with a model of the real system instead of with the actual system itself [33].

Narsingh deo [30] has the following to say about simulation: There is no unifying theory of computer simulation. Learning simulation does not consist of learning a few fundamental theorems and then using them and their corollaries to solve problems. There are no underlying principles guiding the formulation of simulation models. Each application of simulation is ad hoc to a great extent.

Simulation has changed in a fundamental sense, the way in which research is conducted today. Earlier most experiments like wind tunnel, river basin, network analyzers, aircraft flight simulators were conducted physically in laboratories. Today a majority of these experiments are simulated on a computer. "Computer experiments" besides being much faster, cheaper, and easier, frequently provide better insight into the system than laboratory experiments do.

The problems facing industry, commerce, government, and society in general continue to grow in size and complexity. The need for procedures and techniques for better
understanding of the problems and solutions is hard felt. Simulation is one such technique which offers us a better insight into the problem by its modelling technique before a solution is suggested. Simulation is usually described as an art or a soft science, because the useful results of the study depend upon the skill of the modelling team. Korn [27] states that simulation for research and design, education, training, and partial system tests accounts for a substantial fraction of engineering computation. He also points out that simulation can be incomparably cheaper than real experiments. At the present state of the art, there is no scientific theory to guarantee the validity of a simulation process before the experiment is performed. Instead, the suitability of a model is judged by the correspondence of results from the model with known results obtained by observations of systems comparable with the system under examination.

In any simulation study some subsystems may follow a well-known behaviour pattern or lend themselves to direct mathematical analysis. Such systems should not be simulated, but their contribution to the total system must be included in the model.

Kettenis [26] states that depending on the level of detail needed to meet the objectives of the simulation study one may formulate several models of a system — continuous
change models, discrete change models, or combined changed models, linearized or non linearized models and so on.

Simulation models could be employed at four levels [38]:
* As explanatory devices to define a system or problem
* As analysis vehicles to determine critical elements, components and issues
* As design assessors to synthesize and evaluate proposed solutions
* As predictors to forecast and aid in planning future developments

The various stages of development of simulation can be described as follows [38] :

1. Problem Formulation
2. Model Building.
3. Data Acquisition.
4. Model Translation.
5. Verification.
6. Validation.
7. Strategic and Tactical Planning.
8. Experimentation.
9. Analysis of Results.
10. Implementation and Documentation.

Simulation is an iterative process. Analysis of the output will suggest modifications to the strategy, such as
changes in priority or sequencing rules. Thus, step by step, we gain more knowledge about the system and its performance until there is sufficient information to make final recommendations about the system to be implemented.

Simulation is not normally used to find the optimum solution of a problem. The simulation specialist defines alternatives and chooses the one that gives the best performance. It does not follow, however, that this solution is the optimum for the system. Simulation is an excellent technique when other methods fail.

Simulation is the only way to tackle many problems that would otherwise remain unsolved. It is a means of discovering many unsuspected properties of the systems that we study and thus helps us to improve our understanding of these problems.

Each simulation run is a statistical experiment that is time consuming. Generally the objective of the simulation study is to understand and predict the behaviour of a real system, particularly the variation in response when modifications are made either to the input or to the decision rule.

1.2 MODEL

Elzas[10] defines a model to be "A compact representation of a real phenomenon which can generate a behaviour comparable with some behaviour of interest in the
real system and moreover, a vehicle to make evident key characteristics of an object under study."

Models are descriptions of systems [38]. According to Veerapandian [53], a model can be defined as an abstraction of reality or something that looks like the real thing without being the real thing. If a model can be made to look and behave like the real thing, it becomes possible to think of observing the behaviour of the model by subjecting it to various stimuli and interpreting its behavioural response to give a direct insight into the probable behaviour of the reality. It can be stated that simulation is the laboratory of management where dynamic experiments under controlled conditions can be performed and results analysed to yield courses of action.

In the physical sciences, models are usually developed based on theoretical laws and principles. The models may be scaled physical objects (iconic models), mathematical equations and relations (abstract models), or graphical representations (visual models). The modelling of complex, large-scale systems is often more difficult than the modelling of physical systems [37] since human decision making is an integral part of the system.

A basic requirement for any model is that it should describe the system in sufficient detail for the behaviour of
the model to provide valid predictions of the behaviour of the system [4]. The actual process of formulating the model is one which is largely an art [38]. The structure and operating rules of the system are well understood to extract the essence of the system without getting lost into unnecessary details. The model should be easily understood, yet sufficiently complex to realistically reflect the important characteristics of the real system.

1.3 PROGRAMMING SIMULATION PROBLEMS

Developing a simulation program from scratch for each and every problem is tedious and time consuming work. When a few more aspects of the model are to be added or refined, then sometimes the entire structure of the program has to be changed resulting in new design and coding. In most cases, the focus shifts from model building to program writing.

Special purpose simulation languages like SLAM, DYNAMO, SIMSCRIPT, GASP, SIMAN, SIMULA, ACSL, NDTRAN, and GPSS help to reduce the burden of programming to develop a model considerably. The analyst need not get drowned in programming while trying to simulate a model. The simulation language in general outputs additional information along with the model solution which is useful for carrying out consistency checks to determine if the simulation is representing the model being simulated [31]. GPSS is probably the most popular
simulation language; a 1983 survey indicates that 40 percent of the respondents used GPSS for simulation modelling - more than twice as many as the next most frequently used language [29].

GPSS (General Purpose Simulation System) is both a language and a computer program. As a language, it has a well defined vocabulary and grammar with which certain types of system models can be unambiguously described [46]. As a computer program, it interprets a model described in the GPSS language, thereby making it possible to conduct experiments with the model on a computer.

1.4 NEW TRENDS IN PROGRAMMING

The Object Oriented approach is a new way of approaching the job of programming. The powerful ideas of structured programming are combined with new concepts like data abstraction, data encapsulation, inheritance and polymorphism to look at the task of programming in a new light.

The concepts of object oriented programming and the facilities like inheritance open new vistas in the programming environment.

1.5 FUZZY SETS

For factual models or modelling languages two major complications arise [60].
1. Real situations are very often not crisp and deterministic and they cannot be described precisely.

2. The complete description of a real system often would require by far more detailed data than a human being could ever recognise simultaneously, process and understand.

Due to lack of information, the future state of a system might not be known completely. This type of uncertainty has long been handled appropriately by probability theory and statistics and is called stochastic uncertainty.

The vagueness concerning the description of the semantic meaning of the events, phenomena or statements can be described as fuzziness. The notion of a fuzzy set provides a convenient point of departure for the construction of a conceptual framework which parallels in many respects the framework used in the case of ordinary sets, but is more general than the latter and proves to have much wider scope of applicability. Such a framework provides a natural way of dealing with problems in which the source of imprecision is the absence of sharply defined criteria of class membership.

"Imprecision" here is meant in the sense of vagueness rather than the lack of knowledge about the value of a parameter required. Making use of the Fuzzy blocks many systems could be modelled so that the model describes the
problem more realistically especially in areas in which human judgment, evaluation, and decisions are important.

1.6 SURVEY AND CURRENT STATE OF ART

1.6.1 SIMULATING REAL LIFE SYSTEMS

Veerapandian [53] has simulated the movement of the town bus of a particular route in Madras city. This model represents the time taken by the bus to cover each bus stop, time spent by the bus waiting for the proper signals, and also the time taken by the passengers to board and leave the bus. This model was implemented in GPSS.

Sarukesi [45] has simulated the flow line production system to aid management decision making by making use of the general purpose high level language FORTRAN. In this work the emphasis has been given to the decision of allocating resources such as inter operational stock, maintenance facilities and of installing modern equipment/facilities to the production line.

Seila [48] has developed a model for financial risk analysis. This simulation was carried out making use of the software package LOTUS 1-2-3. This method of developing a macro in a software package for simulation is a new approach.

Vivekanandan and Venkata Rao [54] have modelled the freight crew scheduling policies of the vadodara division of...
the Indian Railways. The effect of various personnel policy options on the total number of freight crews to be employed was simulated and analysed. This work was carried out in the special purpose simulation language SLAM - II.

Vivekanandan and Venkatapathy [55] have modelled the city bus movements of routes covering Bharathiar University to study the effect of bus timings on the inter arrival time of the buses to the various stages in routes under consideration. Here again, the software package LOTUS 1-2-3 was used for the simulation.

Vivekanandan et al [56] in their work have used FOXPRO to simulate the arrival of passengers to various bus stops and the movement of buses in those routes.

1.6.2 SIMULATION AND AI

Elzas [12] notes that because there is evidence that modelling and simulation and Artificial Intelligence have basic, operational, elements in common, a number of tools and methods useful for one are probably profitable for both.

Jones [24] observes that an expert monitoring system would be able to make run time decisions as to data collection. The expert monitor can identify when the steady state of the model has been reached and can start collecting the statistics. It can also abort an unproductive simulation.
Sargent [44] concludes in his article that it is going to be extremely difficult to develop an expert system for model validation and a better way to do model validation will not be forthcoming at least any reasonable time frame.

Elzas [13] points out that even when using the best possible modelling methods and the best data available - the reliability of models of societal systems is mostly very poor, and validation often not realistically possible.

Various reasoning mechanisms can assist the modeller in the data analysis task - they can analyse the output data and suggest possible causes of observed bottlenecks for the modeller's consideration [39].

Mellichamp [29] in his work has attempted the integration of expert system methodology in the model development task. He has emphasised through his work that an expert system could be developed which could function as a debugging advisor to the simulation analyst. The expert debugger was used to debug the model developed in GPSS.

Nielsen [34] notes that although the fields of artificial intelligence and simulation are concerned with the development and use of models these fields evolved almost independently until relatively recently. He also notes that the model development experiment demonstrated that the inclusion of Artificial Intelligence techniques in the
developer's tool kit can result in more rapid development, improved user interfaces, and lower required skill levels for users.

Tag Gon [51] in his article describes the progress made in developing the Discrete Event System Specification, a knowledge-based modelling and simulation environment which allows users to create models and store them in its knowledge base. This environment has been developed in the LISP dialect.

Jhyfang Hu [22] in his work presents a new methodology for knowledge acquisition, termed Knowledge Acquisition based on Representation. Instead of treating acquisition and representation separately and sequentially, this method deals with acquisition and representation in an integrated manner.

Jeff Rothenberg [21] observes in his work that recent advances in object-oriented simulation languages, rule-oriented approaches, logic programming and interactive graphics promises to revolutionize simulation, transforming it into something far more powerful, more useful and more believable.

The graphical editor links Object-Oriented Programming and frames with graphics [34].

The AI and Simulation group at the University of Arizona are involved in the development of discrete Event System
Specification (DEVS) in the LISP Environment [51].

An Expert system was developed [23] which could serve as a debugging advisor to the simulation analyst. But the above work does not have a direct interface with the simulation environment. It was pointed out in the paper that the future work could try to give direct interface to the simulation environment.

1.6.3 SIMULATION AND C / C++ LANGUAGES

Zeigler [58] states that advances in simulation environments towards support of hierarchical, modular model construction facilitate synthesis of complex models by hierarchical assembly of modular blocks to form higher levels of building blocks. The class structure of the C++ language naturally helps for this kind of approach.

Davis [6] while developing strategic level war gaming and simulation observes that a language like C which is not usually mentioned in connection with simulation modelling has many features akin to simulation like extensive use of pointers for a centralized data base.

The compactability between Object-Oriented Programming paradigms and discrete event world view formalisms has been well noted [59]. The programming paradigms naturally incorporate AI knowledge representation schemes within
Padmini (35) has developed a SIM Library in the 'C' language to be used for discrete event simulation. The emphasis mostly was on generating variates according to Normal, Poisson, Geometric, Exponential, Gamma, and Erlang distributions.

Jeff Rothenberg [21] has noted the following:

Recent advances in Object-Oriented Simulation Programming, Rule oriented approaches, logic programming, automated inferencing, and interactive graphics are facilitating a new generation of "Knowledge-based" simulation environments. The synergy of these techniques promises to revolutionize simulation, transforming it into something far more powerful, useful and believable.

Ravi Venkatesh [40] has developed software in 'C' which takes input in the form of a graphical representation of a model and generates an equivalent GPSS-PC program.

1.6.4 SIMULATION AND FUZZY SETS

Kaufmann [25] states that the laws of thought that are to be included in the programs of computers are obligatorily formal; laws of thought in the dialogue of man are fuzzy. When software constructed with respect to a fuzzy logic becomes operational and when fuzzy hardware becomes
industrially possible, then man machine communication will be much more convenient, rapid and better adapted to the solution of problems.

Zadeh, Professor of computer science at the University of California at Berkely, laid the foundation of fuzzy logic [16]. At a 1978 tournament, a computer program for backgammon which had taken four years to develop was promptly defeated by its first two human opponents. When the program’s originator revised the system using fuzzy set theory, it succeeded in defeating the world’s backgammon champion.

Zimmermann [60] notes the following on fuzziness:

Human thinking and feeling, in which ideas, pictures, images, and value systems are formed, has certainly more concepts or comprehension than our daily language has words. Fishwick [15] in his work mentions that there is no sharp dividing line between qualitative and quantitative simulation. He predicts that in future we might see commercially available simulation languages that can allow for fuzzy and heuristic model input in addition to the usual quantitative input.

If one considers, in addition, that for a number of notions we use several words then it becomes quite obvious that the power of our thinking and feeling is much higher than the power of a living language. If in turn we compare
the power of a living language with the logical language, then we will find that logic is even poorer.

All traditional logic habitually assumes that precise symbols are being employed. It is therefore not applicable to this terrestrial life but only to an imagined celestial existence.

Fuzziness can be found in many areas of daily life. It is particularly frequent, however, in all areas in which human judgment, evaluation, and decisions are important.

1.7 THE PRESENT WORK

An attempt is made to design the blocks of the historically popular GPSS in the object oriented environment in C++. New data structures and methods available in the OOP are used so that Fuzzy set interfaces could be provided.

It has been observed that incorporating the fuzziness of the real world into simulation has not been taken up so far. In a few cases it was suggested that the future systems may have fuzzy model input. So it was decided to introduce fuzzy input and processing blocks into the popular simulation language GPSS. The study above shows the need for developing facilities available in GPSS in the various environments like DOS, Unix, LAN and providing Fuzzy capabilities to the GPSS. So it was decided to develop software in a language available
in the above environments that will convert the GPSS source into executable code and to include some fuzzy blocks to the GPSS.

1.7.1 MAIN OBJECTIVES OF THIS WORK

The main objective of this work is to design and develop a system in C++ to provide the facilities available in GPSS so that it can be used in any environment like DOS, Unix, and WINDOWS and to incorporate blocks that can represent the fuzziness of the real system to be modelled.

1.7.2 SPECIFIC OBJECTIVES

* To provide software in C++ that converts the GPSS source code into executable code, so that portability of the model is achieved.

* To provide the facility so that the user can develop the model in the C++ environment by identifying each block of GPSS to enable it to give a better readability of the model and help to debug the model quite easily. The entire model should have a set of C++ member functions which offers scope for experimentation with block characteristics enhancement. These member functions interact with similar internal data structures of the previous environment. In the first implementation the
internal data structures are accessed from the program table, whereas in the second they are all accessed from the respective block structure. The approach used in this implementation should eliminate the need to write the grammar, lexical analyser and parser.

* To incorporate the concept of fuzzy logic by the addition blocks like ZENARATE and ZATE so that the fuzziness of some of the real life situations could be modelled. Fuzzy input is given by the block ZENERATE and fuzzy processing is taken care of by the ZATE block.

1.7.3 METHODOLOGY

The model represented by the GPSS block statements should be made available in an ASCII file. An internal data structure called program-table should be created. All the block equivalents of GPSS should be made available as derived classes of this program-table. A pure virtual function should be defined in the base class. The virtual functions in the derived classes should perform the job of the respective blocks. The program table should maintain the various objects' pointers. By making use of these pointers the virtual function could be invoked. Hence this could facilitate to achieve the objectives defined earlier.
The blocks that are represented by the derived classes include Generate, Advance, Queue, Depart, Seize, Release, Enter, Leave, Terminate, Assign, Priority and Transfer.

Various blocks of the GPSS and its general features are discussed in APPENDIX I.

Ten different case studies from Schriber [46] were solved making use of GPSS-PC and the newly developed C++ software and the results are compared. Two problems making use of Fuzzy blocks are also solved.