TITLE

An Artificial Intelligence Framework
for
Experimental Design and Analysis
in
Discrete Event Simulation

AUTHOR

Richard Paul TAYLOR

INSTITUTION and DATE

UNIVERSITY OF WARWICK
1988

An Artificial Intelligence Framework

for

Experimental Design and Analysis

in

Discrete Event Simulation

by

Richard Paul TAYLOR

A Thesis submitted in partial fulfilment
of the requirements for the degree of
"Doctor of Philosophy"

University of Warwick
School of Industrial and Business Studies
July 1988

Table of Contents

Appendices

Appendices

Figures

Tables

Acknowledgements

The successful completion of this research would have been impossible without the contribution of several individuals. Although only a few can be named, I particularly wish to acknowledge the following. First I thank my supervisor, Dr Robert D Hurrion, for all his advice throughout the period of this project.

This thesis benefitted significantly from the assistance of two special individuals. Dr Andrew Flitman spent much time in the initial phases of the research teaching me about his "link" which is used in the early chapters. Marcia Barnes devoted many hours to assist in the preparation of this manuscript. I thank them both for their help and support.

I also gratefully acknowledge the award from the Senate of the University of Warwick which helped to fund this research.

Finally, I am indebted to all the people who kindly gave up hours of their time to test formally and informally the WES system during all of its development stages.

Abstract

Simulation studies cycle through the phases of formulation, programming, verification and validation, experimental design and analysis, and implementation. The work presented has been concerned with developing methods to enhance the practice and support for the experimental design and analysis phase of a study. The investigation focussed on the introduction of Artificial Intelligence (AI) techniques to this phase, where previously there existed little support. The reason for this approach was the realisation that the experimentation process in a simulation study can be broken down into a reasoning component and a control of execution component. In most studies, a user would perform both of these. The involvement of a reasoning process attracted the notion of artificial intelligence or at least the prospective use of its techniques.

After a study into the current state of the art, work began by considering the development of a support system for experimental design and analysis that had human intelligence and machine control of execution. This provided a semi-structured decision-making environment in the form of a controller that requested human input. The controller was made intelligent when it was linked to a non-procedural (PROLOG) program that provided remote intelligent input from either the user or default heuristics. The intelligent controller was found to enhance simulation experimentation because it ensures that all the steps in the experimental design and analysis phase take place and receive appropriate input.

The next stage was to adopt the view that simulation experimental design and analysis may be enhanced through a system that had machine intelligence but expected human control of execution. This provided the framework of an advisor that adopted a consultation expert system paradigm. Users were advised on how to perform simulation experimentation. Default reasoning strategies were implemented to provide the system with advisory capabilities in the tasks of prediction, evaluation, comparison, sensitivity analysis, transient behaviour, functional relations, optimisation.

Later the controller and the advisor were linked to provide an integrated system with both machine intelligence and machine control of execution. User involvement in the experimentation process was reduced considerably as support was provided in both the reasoning and control of execution aspects. Additionally, this integrated system supports facilities for refinement purposes that aim at turning the system's knowledge into expertise. It became theoretically possible for other simulation experts to teach the system or experiment with their own rules and knowledge.

The following stage considered making the knowledge of the system available to the user, thereby turning the system into a teacher and providing pedagogical support. Teaching was introduced through explanation and demonstration. The explanation facility used a mixed approach : it combined a first time response explanation facility to "how" and "why" questions with a menu driven information system facility for "explain" requests or further queries. The demonstration facility offers tutorials on the use of the system and how to carry out an investigation of any of the tasks that the system can address.

The final part of the research was to collect some empirical results about the performance of the system. Some experiments were performed retroactively on existing studies. The system was also linked to a data-driven simulation package that permitted evaluation using some large scale industrial applications. The system's performance was measured by its ability to perform as well as students with simulation knowledge but not necessarily expertise. The system was also found to assist the user with little or no simulation knowledge to perform as well as students with knowledge.

This study represents the first practical attempts to use the expert system framework to model the processes involved in simulation experimentation. The framework described in this thesis has been implemented as a prototype advisory system called WES (Warwick Expert Simulator). The thesis concludes that the framework proposed is robust for this purpose.

Chapter 1

Some Considerations about Experimental Design and Analysis

1.1 Introduction

The appeal of simulation to analysts and end-users has been increased, in recent years, by the appearance of tools which provide the facility to accelerate the passage through some of the stages in the development cycle of a simulation model. These stages are :

- formulation
- programming
- verification
- validation
- experimental design and analysis
- implementation.

The first four stages have benefitted from most of the research. Natural Language Understanding Systems (NLUS) assist in formulation (Balmer and Paul 1986). Program generators perform much of the required programming (Mathewson 1985). Visual Interactive Simulation (VIS) has made verification easier (Hurrion and Secker 1978). Recent work by Rao and Sargent (1988) has addressed the issue of validation with an Operational Validity Advisory System (OVAS).

Despite the fact that many languages, eg. SIMAN (Pegden, 1984), now have powerful facilities for collecting and presenting output, there remained until recently a distinct lack of support for the simulation experimental design and analysis phase of a project. Contrasting with the amount of research on the earlier phases, this paucity of support has led to concern about the effectiveness of simulation tools in the hands of non-experts. Such concern has been reflected by Smith (1986) when he highlighted the increasing dangers in the misuse of simulation tools. Smith suggested that with the advent of visual interactive methods, attention has been focused on

the visual interactive side of simulation at the expense of properly designed simulation experimentation. End-users of simulation packages, concluded Smith, are becoming if not unaware of traditional statistical simulation problems, for example considerations of steady states or variance reduction methods, then unable to design simulation experiments which will produce results that are statistically significant. Also many end-users may not be giving a valid interpretation to their simulation results. This is particularly true of interactive methods where interactions at run-time may destroy any notion of a steady state. O'Keefe (1986a) stressed the growing need to support the use and guard against the misuse of simulation : a task which is particularly difficult when estimates of over 700 hours of formal tuition are needed to give someone the basic tools of simulation (Shannon 1986).

Some research has been started recently in the area of experimental design and analysis. Luker (1986b) and Haddock (1987) have developed support in this area in the context of program generators. O'Keefe (1986a, 1986b) developed a prototype to assist with transaction flow model experimentation. In the context of knowledge-based simulations, Reddy, Fox and Hussain (1985) have considered automating the analysis of simulations.

It is suggested that the traditional human thought-processes (generated by simulation knowledge and experience) and activities carried out by the analyst during the phases of experimental design and analysis can be viewed as the combination of intelligent (non-procedural) reasoning and formal (procedural) control of execution. The reasoning aspect appears in the form of intelligent specification of the experiment to be carried out. The control aspect deals with executing the experiment in accordance with the specification obtained from the intelligent component.

If intelligent reasoning and control (of execution) are considered as the two complementary processes for experimental design and analysis, it is possible to consider the potential and feasibility of providing a support environment that assists or replaces the human nature of either (or both) of these processes in the development of a system that gives a machine/artificial nature to the intelligent reasoning or control (of execution) processes. Such systems would provide a support environment for the phases of experimental design and analysis.

If the execution of experiments can be controlled by a (computer) system, then the risk of error or omission regarding the appropriate steps to be carried out is reduced or eliminated.

If the traditional thought-processes required in the experimental design and analysis phase can be modelled using artificial intelligence (AI) techniques, then the human reasoning process can be assisted or replaced. If a modular form (eg. production rules) is adopted to represent the knowledge, then the incremental development of advisory systems in experimental design and analysis is possible. Such a system could greatly increase the understanding and confidence of users of simulation if the captured knowledge could be made available to the user through an explanation or teaching facility.

Further advantages of developing an AI framework for simulation experimentation may result if the human involvement is replaced in both control and intelligent reasoning and processes leaving an integrated system. The system could not only learn from, explain to or teach the user, but also act as his decision-maker in the execution of simulation experiments to address a specified goal.

1.2 Research Objectives

This PhD thesis aims to investigate the feasibility and suitability of AI techniques to the development and integration of advisory systems for simulation experimentation and analysis. These systems would assist or replace the human involvement in both the intelligent and control aspects of experimentation.

The feasibility of replacing the human component in the control of experiments could be achieved by considering the physical processes involved in experimentation, and by developing a procedural system which performed the steps identified. Such a system would be a rule/human-assisted expert which sets up, runs, controls and monitors simulation experiments, whilst seeking intelligent input when required from human origin or rules saved in AI form. This would also suggest the suitability of AI techniques to storing simulation experimental design and analysis knowledge.

It is then anticipated that the appropriateness of AI techniques to simulation experimental design and analysis could be further established by the development of a non-procedural 'consultation-based' expert system to address specific issues of simulation usage. These would be problems of prediction, evaluation, comparison, sensitivity analysis, optimisation, transient behaviour and functional relations. The original objective of supporting the experimentation process could be seen as achieved by observing a user with minimal simulation knowledge and equipped with a verified model being able to address complex problems with the satisfaction of producing results which are significant, and on which he can base decisions.

Having developed a support environment for experimental design and analysis, work is then carried out to extend the environment with explanation and teaching facilities. This now allows a user to ask "why?", "how?" and "explain" as well as the more normal "what-if?" questions of a simulation project. It is seen how, by reason of the modular nature of its expertise, the system can either learn from experienced analysts or teach users with less knowledge.

The thesis is to conclude that artificial intelligence techniques may successfully be used to support the experimental design and analysis phase of a simulation study.

1.3 Overview of Chapters

In chapter 2, the rationale of the approach is uncovered through a literature review of the current practices relating to Simulation, Decision Support Systems, Artificial Intelligence, Expert Systems, Heuristics, Experimentation, and the applications of Artificial Intelligence to Simulation. It is seen that simulation experimentation requires expertise which previously was considered to be the domain of humans. Now, the use of heuristics within an expert system framework appears to provide a means of incorporating this expertise. Having considered the literature, an argument is put forward to support the value of the research undertaken in this thesis.

The third chapter covers the investigation into the suitability of AI techniques to support the development of an intelligent Controller. This system has human intelligence but machine control. It proposes a formalisation of the experimentation process by adopting a procedural framework. It directs the user to proceed sequentially through a number of predefined steps in designing his experiment, thus insuring that all relevant issues are addressed. The controller then automatically carries out the execution of the designed experiment. A means of separating the procedural control from the non-procedural input is presented using two machines linked via serial RS232 interface. This framework offers the possibility of integrating machine intelligence into the Controller process, to help the user proceed through the different steps of the specification procedure.

In chapter 4, the appropriateness of the expert systems framework for a simulation experimentation advisor is considered, leading to the development of an Advisor that has machine intelligence but requires human control of execution. Through questioning the user, the system gathers information about the simulation model under consideration. An initial experiment is devised using production rules according to the objective of the study. The design of any experiment considers issues about how to achieve a steady state and how to obtain statistically significant results. The analysis of the first experiment may prompt the advisory system to refine the experiment or request further experiments before it can yield a sensible conclusion. An account of the implementation of this system in Prolog is given with a detailed look at the experimentation

problems that may be addressed using the system.

Chapter 5 looks at the linking of the Controller to the Advisor in order to produce a system which has both machine intelligence and machine control of execution. The link permits the system to automatically retrieve some understanding about the components of the simulation model under consideration. Such a system permits a user to specify through a consultation process an experimentation goal to be achieved. This integrated system performs intelligent reasoning to design appropriate experiments and then controls their execution.

The sixth chapter considers the necessary features to provide teaching capabilities to the system. The objective is to enhance human understanding of the experimental design and analysis process through the use of machine knowledge. Two complementary approaches to teaching are considered: explanation and demonstration. The issue does not surround the pedagogical performance of these teaching mechanisms but rather the extraction of the concentration of knowledge, that has been given to the system, for the benefit of the user.

In chapter 7, some results and considerations about the use and application of the developed integrated system are presented. By performing a number of retro-active studies, the system is seen to overcome a number of problems relating to bad data transcriptions, bad design, steady state issues witnessed in the review of 59 student projects. The application of the system to a data-driven simulation package indicates the robustness of the proposed framework to be transported to other simulation environments. The ability of the system is judged to have attained student level by performing as well, in addressing an optimisation problem, as a class of students who had simulation knowledge but not necessarily expertise. The framework was seen to support further advanced issues such as the handling of uncertainty which increases the quality of its reasoning process. An eighty hour long experiment involving twenty users with minimal simulation knowledge concludes that the system significantly helped these subjects achieve simulation studies comparable to the class of students considered.

Chapter 8 concludes the thesis with a summary of the research and a discussion of results, conclusions and indications towards future research. The artificial intelligence framework pro-

posed in this thesis is concluded to be valid to support the experimental design and analysis phase of a simulation study.

End of Chapter

Chapter 2

Current Practices in Simulation and Artificial Intelligence

2.1 Introduction

As its title suggests, the essence of this thesis lies in the integration of artificial intelligence techniques into the field of discrete event simulation. The literature study undertaken in this chapter will attempt to explain how artificial intelligence techniques have come to find a place in the evolution of discrete event simulation. The objective is twofold. Firstly, to give contextual meaning to the problem expressed in chapter 1 : the apparent lack of support for experimental design and analysis in simulation experimentation. Secondly, to provide an introduction to the tools and concepts that were mentioned in the previous chapter. Because these tools and concepts come from different disciplines, the study will necessarily be diverse.

The role of discrete event simulation within Operational Research (OR) is considered first (section 2.2). The phases of a simulation study are described and developments which enhanced these different phases are mentioned. The need to eventually look outside the traditional world of OR is explained by the type of problems which analysts encounter. These problems are described and paralleled to those which appear in the world of decision-support systems (section 2.3). A detailed look is given to the emerging tools of artificial intelligence (section 2.4), and how they might be of use to simulation analysts. Finally, a review of the current work on the integration of AI techniques into simulation is undertaken (section 2.5), before a conclusion is reached supporting the research contained in this thesis (section 2.6).

2.2 Simulation and Operational Research

2.2.1 The Simulation Approach

> *"Operational Research is the application of the scientific method to assist in the solution of problems which occur in the management of an organisation. Its purpose is to provide quantitative measures on which decisions about the operation of the organisation and changes to its resources and structure can be based. The methods used to obtain these quantitative measures are known as the techniques of Operational Research. One, among many, of these techniques is Simulation"* (Clementson, 1980).

Simulation can be broadly described as the technique of problem solving by the observation of the performance over time of a dynamic model of the system (Gordon, 1978).

Each of the techniques of Operational Research is based on one single central idea : a model. A model is an abstraction of the system under study in which the essential structure has been retained, while the extraneous detail has been omitted (Pritsker, 1984) . One essential feature of any model is its ability to be manipulated.

(I) Simulation Models

There are several types of simulation models (Solomon, 1983; Clementson, 1980):

a. Physical models used for training purposes : eg simulators for training car drivers, pilots or astronauts.

b. Business games ; often called open simulations. These are models of an organisation used to instruct potential or trainee managers to manage their organisation.

c. Continuous simulation : these are built on digital or analogue computers which integrate differential equations. The most popular application areas for this type of technique are the simulation of physical, chemical and atomic processes. A minor modification of this technique is the use of difference (rather than differential) equations. This is the basis of "Industrial Dynamics" and the other financial simulation techniques.

d. Discrete simulation. These models are so called because they consider time and almost all other variables to be discrete. The concern of this thesis lies with discrete event simulation.

(ii) The Simulation Approach

(Discrete Event) Simulation differs significantly from other models and techniques. These other techniques will permit a problem to be solved analytically, subject to a number of basic assumptions about the problem environment. Many of the assumptions required to model problems, so that they can be solved analytically, are not required in simulation: thus larger and more complex systems can be studied. In many situations simulation is the only viable means for analysis (Gordon, 1978).

There are differences, as illustrated by figure 2.1, between the simulation and analytic approach to the modelling solution process.

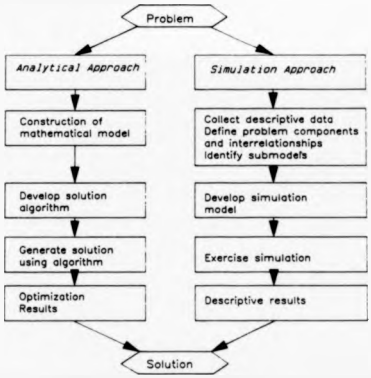


FIGURE 2.1 : Simulation Versus Analytic Approach to the Modelling Solution Process (Davis & McKeown, 1984)

The analytical approach stresses the formulation and development of mathematical models and the analytical or mathematical solution to the models. In most cases analytical solutions are in the form of algorithms which yield "optimal" solutions. Simulation, on the other hand, is a descriptive modeling process. The modelling process associated with simulation generally

involves collecting data which describe input and operational factors and defining the inter-relationship of the factors (variables), inputs, and other components of the problem being studied. The output from a simulation model is in the form of performance descriptors. By exercising the models, the characteristics of the problem can be explored (Davis & McKeown, 1984). It should be noted that although simulation output is always descriptive in nature, a "search routine" can be included in the simulation model to provide an optimal or near optimal solution. The term "near optimal" is used because the solution may be optimal in terms of the defined model, but this does not guarantee the solution is a global optimal. Optimality in simulation thus may be an approximation of optimality which occurs in mathematical programming (Davis & McKeown, 1984).

In consequence, the models of discrete event simulation possess certain features which distinguishes them from other types of models. The main features are ennumerated by Clementson (1980) :

- Predictive (as opposed to optimising)
- Random (it represents the stochastic elements of the system)
- Empirical (as opposed to analytical)
- Discrete (as opposed to continuous)
- Closed (the model works without intervention)
- Time-dependent (not static)

2.2.2 The OR Analyst and Computer Simulation

Modern computers allow the analysts to explore the whole range of feasible options in a decision-problem (Solomon, 1983). Indeed, while it is possible to solve many complex problems using hand simulation, this can be very time consuming. Also Poole & Szymankiewicz (1977) point out that *"simulation is ideally suited for computers as the rules for operating a model are repetitive but simple in concept"*. Indeed for discrete systems, where the prime interests are in the events, the equations are essentially logical, stating the conditions for an event to occur. This thesis is concerned with computer simulation.

Because realistic computer simulation may require long computer programs which are time consuming to develop, Pidd (1984) argues that in *"one way, computer simulation should be*

regarded as a last resort." The OR analyst resorts to simulation when it is either impossibly or not economically feasible, or just sufficiently inconvenient to experiment with and study the real system (Conway & al, 1959). Simulation does offer the following advantages as compared to direct experimentation:

- cost : real experiments may be very expensive as compared to the cost of developing a simulation model (Bobilier & al, 1976);
- time : although development of a model may take time, it allows simulation of week, months or even years of operations in seconds of computer time (Fishman, 1973);
- replication : the real world rarely allows precise replication of experiments in management science. *"It seems unlikely that an organisation's competitors will sit idlely by as a whole variety of pricing policies are attempted in a bid to find the best* (Pidd, 1984)." Simulations are precisely repeatable.
- safety : simulation studies permit investigation of extreme conditions where it might be dangerous to do so in real life. *"Simulated aircraft causes little damage when it runs out of fuel in the simulated sky* (Pidd, 1984)".

" Computer simulation may well be regarded as the last resort. Despite this, it is surprising how often such an approach is needed" (Pidd, 1984).

Computer simulation methods have developed since the early 1960's. The advancement and acceptance of simulation as an OR tool has been directly tied to the developments in both computer hardware and software.

(I) Hardware for Simulation

On the hardware side, Shannon argues (1986) that the increasing power and decreasing costs have made computers available to organisations of all sizes. The availability of powerful microcomputers makes the use of simulation feasible for even the smallest organisational unit. The importance of this fact to Operational Research was stressed by Ranyard & Blewitt (1983) :

> *"Personal computing allows models to be manipulated directly by the manager/decision maker. This form of computing is far more appropriate for good OR work, where the objective is to persuade the manager to view his decision and explore alternatives in terms of the model, not to provide a single correct 'answer'. Consequently, batch programs returning a single result were of limited use to OR, whereas personal computing can embody the developed model in a form which is real to the manager."*

Balmer & Paul (1986) reiterated this fact. *"Microcomputers have allowed the modeller literally to take the model, rather than simply a pile of computer output, to the user for verification"*. At Warwick university, Hurrion (1981) demonstrated the use of Visual Interactive Simulation (VIS) on a microcomputer.

In the light of the above , it was decided that the work of this thesis should be microcomputer based.

(ii) Software for Simulation

The development of simulation software has played an equal role in advancing the state-of-the-art. In this rapid overview (section 2.2.5 considers some points in more detail), it can be shown as argued by Nance (1984), that the development of simulation software can be divided into five periods:

- Pre - 1960, all simulations were written in general purpose languages such as FORTRAN.
- 1960 - 1965, the first special purpose simulation languages appeared : GPSS, SIMULA, GASP, SIMSCRIPT, CSL.
- 1966 - 1970 saw the introduction of the second generation of these simulation languages. GPSS II, III, 360 and V were developed as were SIMSCRIPT II, II.5 and II-plus. SIMULA 67, Extended CSL (ECSL) and GASP II replaced earlier versions.
- 1971 -1980 witnessed the introduction of new languages such as SLAM and Q-GERT and the addition of new features to existing languages. GASP IV added the ability to combine continuous and discrete components in the same model, while SIMSCRIPT II.5 incorporated the process concept and also added a continuous capability. GPSS/H corrected many of the

obvious shortcomings on earlier versions of that language and introduced a greater degree of user interaction.

Towards the end of this fourth period, attention began to shift to a concern for easier model development and execution. Several authors began to discuss potential methods for easier programming (Mathewson, 1974; Clementson, 1980). Others began to take a more formalised approach to the modelling process (Zeigler, 1976; Kleine, 1977).

In 1979, Oren & Zeigler produced a significant paper in which they pointed out several weaknesses in current languages and proposed developing new languages based on two concepts. First simulation languages should functionally separate the logically distinct stages of modelling, experiment design and output analysis into separate activities. Secondly, simulation environments should be created to take advantage of current computer capabilities for data base management, graphics and program verification.

1980 - 1988 The current phase is one in which the development of simulation software is in a significant transition period. Section 2.5.3 reviews in greater detail the current research. However, the emphasis is upon ease of use and providing an "integrated simulation environment" rather than simply more powerful languages (Shannon, 1986). Henricksen (1983) defines an integrated simulation environment to be *"a collection of software tools for designing, writing and validating models; writing and verifying simulation programs (implementing models), preparing model input data, analysing model output data; designing and carrying out experiments with models"*.

2.2.3 Phases of a Simulation Study

A simulation study will go through the following phases (Mize and Cox, 1968):

1. Formulation
2. Programming
3. Verification & Validation

4. Experimental Design & Analysis

5. Implementation

(I). Formulation

a. Purpose: The first step is to clearly and completely specify the purpose for which the investigation is being conducted. An important part of defining the purpose of a study is the specification of criteria by which results of the investigation will be measured (Mize & Cox, 1968).

b. System description: After its purpose has been defined, the system must be described in terms which are amenable to some rather general form of model construction and ultimately to a solution method. Mize & Cox (1968) argue that all systems can be described in terms of the following common features :

1. Components (entities which are independently identified, eg machines);

2. Variables (both external and internal)

3. Parameters (attributes which do not change during the simulation);

4. Relationships (both cause and effect among components and variables.

c. Data collection: The experimenter must determine and specify which data are needed for estimating (Mize & Cox, 1968; Gordon, 1978; Bobilier & al, 1976):

1. The parameters of the system (population parameters, maximum queue length, etc.);

2. The behaviour of the variables of the system (probability distributions, etc.);

3. The nature of relationships in the system (Mize & Cox, 1968; Gordon, 1978).

According to Maisel and Gnugnoli (1972), there are three major sources to obtain this data:

- systems records (historical data), which are normally the most useful, although possibly difficult to obtain;

- expert opinion, which may be easy to obtain, but may suffer the risk of being biassed;

- field studies (data-recording systems), which can be expensive and time-consuming and therefore require appropriate design to ensure their validity.

d. Data processing Once the data has been collected, it will need to be processed and analysed to allow formulation of a model which realistically imitates the system being studied.

(ii). Programming

By the end of the formulation stage, a model will have been obtained which establishes the specifications for what must be programmed. There are several methods of programming simulation.

- Standard package : these are available to cover simulations of common application (eg stock control). The problem with this method is that it offers little flexibility in the formulation of the model ; indeed, it must be acceptable to the package (Poole & Szymankiewicz, 1977).
- General Computer Language : these are traditionally BASIC, FORTRAN, ALGOL, PASCAL. This method will involve heavy programming (Tocher, 1965).
- Simulation Languages : these have been specifically developed for simulation models. The best known, as previously mentioned, are GPSS and SIMSCRIPT, also GASP, DYNAMO, SIMULA, SIMON, ECSL (Solomon, 1983).
- Program Generators : these are a more recent development whereby a model is generated and a program compiled in response to the answers to a series of questions regarding logic, priorities, arithmetic, recording and initial conditions. For example : DRAFT (Mathewson, 1974), CAPS (Clementson, 1974).

Very early on, Tocher (1965) concluded that deciding which method to adopt is most likely to be a question of availability.

(iii). Validation and Verification

In order to draw inferences about the real system from results obtained from the simulated system, the simulation model must be a "reasonably" valid representation of the real system (Pritsker, 1984). A computer simulation model is considered valid if it produces results which are very close to the results which would be produced by the real world system, which the computer model is supposed to represent (Mize & Cox, 1968).

The validity of a model under all variations and all conditions cannot be verified (Maisel & Gnugnoli, 1972). However some assurance of validity can be provided by a demonstration that for at least one alternative version of the simulated system and one set of conditions, the simulator produced results which are not inconsistent with the known performance of the real system. This kind of test is essentially a null test. A model which failed to pass would be exceedingly suspect, but no strong statement can be made for a model which passed (Conway & al, 1959).

Verification is just another aspect associated with validating a model. Verification can be considered equivalent to debugging a computer program and consists of checking to see if the final computer simulation program is equivalent to the simulation model specified (Bobilier & al, 1976). There are a number of tests which can be carried out :

- trace facilities which consist of tracing the movement of all entities within the model for logical correctness (Bulgren, 1982);
- replacing all the stochastic elements with constant values (usually the average). It is then possible to calculate the result of a computer run (Bulgren, 1982);
- starting the model from extreme conditions to see if it behaves as expected;
- by using possibly a simple model, results could be compared with theoretical results obtained from queueing theory (Bobilier & al, 1976).

(iv). Experimental Design and Analysis

a. Introduction

Although the title of this thesis contains the words "Experimental Design and Analysis", it is not the objective to discus and review all existing approaches to this topic. As will be argued, the interest lies in the processes that these approaches may imply to the simulation user. Hence, the work focuses on a framework-development to support these processes which may later, in turn, be used to implement many of the existing approaches. The purpose of this section is thus to present the issues relating to the design and analysis of experiments but not to present the details as to how they are addressed. Where techniques are used, they will be discussed at implementation stage. The complexity of these procedures should become apparent by the diversity of the approaches.

Most investigation by simulation can be resolved into a comparison of alternatives when the model is run under a variety of conditions. (Conway & al, 1959). An important and often difficult problem in simulation concerns the very basic question of the measurement to be made: what is to be measured and when. This is obviously critical for the analysis of results. Usually the analyst is concerned with estimating the state probability distribution of a permanent entity's attribute. For example, one could simply keep track of the amount of time spent in each of the possible system states. The probability of being in a given state is then estimated by the ratio of the time spent in the state to the total length of the run (Conway, 1963).

How often one takes measurements is also critical to the design of experiments. This process of replication is critical in asserting the significance of results. In fact, obtaining simulation results requires addressing a number of statistical issues such as ensuring that observations are independent and identically distributed. *"A simulation is a computer-based statistical sampling experiment* (Law and Kelton, 1982)". Fortunately, Pritsker (1984) notes that most of the statistical techniques are not used significantly differently in simulation studies from their use in other areas.

b. Initial bias and starting conditions

A comparison of two alternatives should be based upon measurements obtained from the steady-state operation of each alternative. Unfortunately, almost without exception, such conditions are attained only after some preliminary running time during which the transient conditions induced by the artificiality of starting are allowed to decay (Conway & al, 1959).

Although, it is not difficult to qualitatively describe the desired equilibrium condition, actually specifying a quantitative measure, which will indicate when it is attained, is more of a problem. Equilibrium implies only that the long-run mean of the process be stationary, it does not deny the existence of runs and cycles, nor does it require that the sample observations be normally distributed (Conway & al, 1959).

Associated with the problem of run-in period, there is also the business of starting conditions. The problems of determining how to start the model, and how to obtain measurements which are not biased by the method of starting or stopping are among the most difficult procedural questions to answer (Conway, 1963). The investigator has at least the following three choices with respect to starting conditions (Pritsker, 1984):

- test each system starting empty and idle,
- test each system with its own "reasonable" starting conditions,
- test each system using a common set of starting conditions which is essentially a compromise between the two different sets of reasonable starting conditions.

Another approach is to use truncation procedures: this is to decide at which point in time sampling should be initiated. Conway (1963), Fishman (1973), Schriber (1974), Gordon (1978) and Gafarian & al (1977) each propose their own rules about truncation. Law and Carson (1979) propose a sequential procedure for determining the length of a steady state simulation. Others propose tests for detecting the presence of initialisation bias using statistical hypotheses concerning the mean of output processes. Schruben & al (1983) test the observed output is consistent with a null hypothesis that the mean of the process did not change throughout a run. Other tests are found in Schruben (1982). Grassman (1982) analysed the estimation error due to initial bias.

And Beal (1982) borrows from the theory of convergence in distribution to develop a criterion for convergence to a steady-state.

c. How to sample and Length of runs

Kleijnen (1977) discusses statistical techniques to address these issues. He notes the distinction between terminating systems and non-terminating systems. In a terminating system, the simulation ends if a specified event occurs. Replication is the only way to increase sample size. This also facilitates analysis as replication yields independent observations. In non-terminating systems, there is no such critical event to indicate the end of a run: the system continues indefinitely. By using a prolonged run, three approaches are possible: nearly independent subruns (batch means), estimation of serial correlations (very theoretical) (Fishman, 1973) or independent regenerated blocks.

d. Number of runs

A suggestion for the number of replications "n" required in order to estimate statistical parameters with desired levels of accuracy is provided by Mize & Cox (1968). Consider the use of a parameter "d" denoting the precision with which the sample variance S-squared is representative of the population variance Sigma-squared. Then the necessary sample size "n" is

$$n = 1 + \frac{2(S)^2}{d^2}$$

For example, if an estimate is required of sigma-squared within 5% of its true value and with a 95% confidence, then

$$n = 1 + \frac{2(1.96)^2}{(0.05)^2} = 3074$$

e. Variance Reduction techniques

When results are collected they need to be processed so as obtain aggregated results that represent the average values of a given measure. Associated with this mean figure must be a consideration

of its variance. Wilson (1982) indicated clearly, by considering numerical examples as above, the limitation of simply increasing the sample size of observations as a variance reduction technique. Other techniques are commonly the use of antithetic random number streams, or the use of common random number (Heikes & al, 1976 ; Wilson, 1982). In a more limited application area, Adkins (1982) proposed a technique for mean flow time in open queueing network simulations.

f. Analysis

Much of the analysis relies on the application of the Central Limit Theorems (Schruben and Goldsman 1982). At the very least, means are computed for which confidence intervals are built. Amer (1982) discusses how to provide statistical confidence in the evaluation of alternatives. This can however lead to coverage errors for confidence intervals: this has been studied by Glynn (1982). Mamrak and Amer (1980) proposed the construction of confidence intervals for proportion estimates instead of means as this approach does not rely on distributions. The analysis of output is complex: data must be tested and the appropriate test is dependent on the characterisation of the data. Bulgren (1982) offers a table to assist (see figure 2.2). He proposes for multivariate analysis the use of techniques like Analysis Of Variance (ANOVA) and regression analysis (see also Kleijnen 1981). However, several authors suggest that the significance level is inappropriate as a measure for large bodies of data as are found in multiple simulation runs because the null hypothesis is too easily rejected (Schwarz 1978, Leonard 1979). They offer alternative tests that are based on likelihood ratios.

In the analysis of multivariate output, there is a problem of making a choice among alternatives. Kleijnen (1975) and Dudewicz (1977) discuss the ranking and selection techniques. Gupta and Hsu (1984) presented a package that implements some of these approaches. They also use notions of indifference zones to group alternatives that are statistically equivalent. Friedman (1984a) also reviewed multivariate simulation output analysis and proposed (1984b) the multivariate general linear metamodel to assist in establishing functional relationships in multiple response simulations.

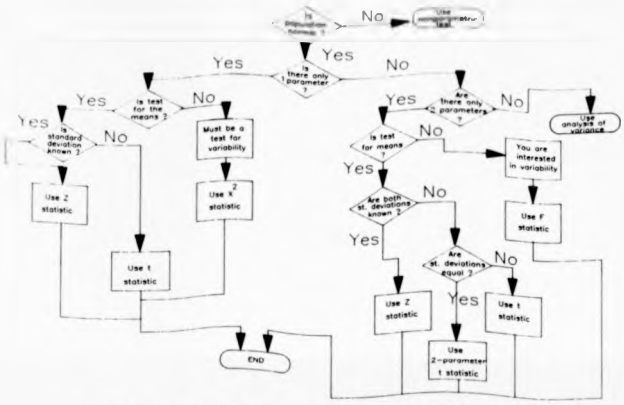


Figure 2.2 : Table to assist with selecting appropriate statistical test

g. Conclusion

This overview of experimental design and analysis issues suggests that this stage of a simulation study is still unresolved and very complicated.

(v). Implementation

This phase is concerned with the implementation of results and recommendation which derive from the analysis of the simulated runs. Although, the results from the simulation model should have established the alternatives which can be applied to the real system, carrying out implementation is often treated by the analyst as the management's responsibility. Though it may be true that implementation is the responsibility of the management, preparation for implementation should equally be the responsibility of those who carry out the simulation study (Mize & Cox, 1968). Such preparation could take the form of a report carefully summarising the simulation study, the experiments and the results.

2.2.4 Approaches to Simulation Modelling

(i) The three-phase approach

Several underlying methodologies to simulation modelling exist. The distinction between these alternative modelling approaches lies in the "next event" selection and timing management. (Hooper, 1986a). The three-phase approach developed back in the late fifties by Tocher (1963) is still the most favoured in the UK (Crookes, 1982). This approach considers two types of occurrences :

- bound activities : which are predictable and hence can be scheduled;
- conditional activities : which are dependent upon certain conditions.

The rules of the three-phase method of Tocher's approach are :

1. A-phase : time scan. Find the next time at which one or more bound activities are scheduled to be executed and advance the clock.
2. B-phase : Execution of bound activities found necessary in the previous phase.
3. C-phase : Testing (and execution of the action part) of each conditional activity.

The executive simulation program cycles round these three-phases until termination time. Termination time is given by a preset duration or when a predetermined terminating condition is reached.

Crookes (1982), O'Keefe and Davies (1983), Balmer & Paul (1986) and Crookes & al (1986) show examples in the current literature of the continuing use of this approach. Their criterion is the ease in modification of the model. This is due to the relationship between independent conditional rules and the three-phase method. This allows for modular programs to be written making the model easier to analyse, comprehend and extend.

(ii). Event-Scheduling

The Event-Scheduling approach involves a succession of unconditional events over time.

The event-scheduling time control procedure selects, from the event list, the event notice having the earliest occurrence, updates the simulation clock to that time, and invokes the corresponding event routine. Any condition testing, other than on clock time must occur within event routines. Events are chosen and processed successively until termination time. (Hooper, 1986a; Laski, 1965; Gordon,1978; Fishman, 1973).

(iii). Activity-Scanning

The Activity-Scanning approach chooses the next event based on scheduled time and condition testing. The basic concept is an "activity" which is (conceptually) a system state transition requiring a period of time. An activity is usually represented as two distinct events which mark the beginning and the end of the activity. The activity-scanning time control procedure scans activities in priority order for time eligibility and other activation conditions, executing the activity routine of the first component whose activation conditions are met. When an activation occurs, the scan starts over again in priority order. This process continues until termination time. (Hooper, 1986b; Laski, 1965; Gordon,1978; Fishman, 1973).

(iv). Process-Interaction

This approach has characteristics related to both the event-scheduling and activity-scanning approaches. Components of a system progress through a sequence of steps (referred to as a process). Each step may consist of a condition segment and an action segment. Execution of the condition segment determines whether execution of the action segment may occur.

The process-interaction time control procedure has two event lists : a future event list (FEL) containing event notices for events scheduled for execution at a later check time, and a current event list (CEL) containing event notices for events which are already eligible from the stand point of time to be executed, but whose other conditions may not yet have been met. Each event notice contains an indication of its components current step location in a process. When time is advanced, all events scheduled for current time are moved from FEL to CEL. Then a

CEL scan occurs. This consists of evaluating each entry's condition routine to determine whether the corresponding components may move to the next step. If so, the step's action segment is executed. A component moves through as many successive steps as possible (ie as long as time need not advance, and condition segments are found to be true). When a component "stops" (due to time or other conditions), the scan resumes with the next CEL entry. When no CEL component can move time is advanced. (Hooper, 1986a; Gordon, 1978; Fishman, 1973; Zeigler, 1976).

(v). Cellular Simulation

The methodology of cellular simulation is described in Spinelli & Crookes (1976). The basic idea of cellular simulation is to split the simulation into non-overlapping activity groups (or cells). Each cell can then be considered as a simulation in its own right.

Computational efficiency is the main advantage of this approach. Indeed, in terms of the three-phase model, a great saving in the execution time is possible, because there will be no testing for a C-activity within a cell unless that cell has had a B-activity executed since the last time the clock was stopped or an elememt has entered or left the cell. This cuts down substantially on the checking of those activity tests which must fail.

Spinelli and Crookes (1976) also indicate the possibility of modelling large systems in separate cells, written at different times and for different purposes and later combined.

2.2.5 Enhancements to the practice of Simulation

Computer simulation is not trivial. There are many difficulties (see Conway & al, 1959; Conway, 1963). Realistic simulation may require long computer programs of some complexity. Simulation verification, validation and experimentation are not easy either. (Pidd, 1984). During the history of simulation, there have been a series of developments to enhance the practice of simulation and to facilitate its usage.

(i). Languages

By 1965, Tocher was able to review a large number of different simulation languages. Most of these were dedicated to a specific machine only and Tocher concluded that the choice of a simulation language would most likely be made by the type of machine available to the user.

The main conclusion from his detailed comparison of these languages was that for occasional use, a simple language, which is easy to understand and learn may be more valuable than one of the sophisticated languages that has many facilities but are much more complicated to use and understand.

Since this review, the number of simulation *"purporting to aid the unwary user "* has greatly increased (Crookes, 1982). Crookes counted 137 such languages. Although concerned by this large number, he noted a *"level of model verification not previously attainable by the use of practical outputs from a running simulation"* (a major benefit of Visual Interactive Simulation (see later)).

In line with the increased use of microcomputers, another contribution to the popularisation of simulation has been the great number of simulation languages and packages implemented on microcomputers. (O'Keefe, 1984).

(ii). Packages

(a). Program Generators

Program generators are probably the most obvious example of packages designed to aid the simulation analysts. The best known are CAPS (Clementson, 1974) and DRAFT (Mathewson, 1974). O'Keefe (1984) put forward the following advantages of program generators :

- they free the model builder from a considerable amount of programming;
- the generator enforces a particular modelling approach and can catch errors and inconsistences in using that approach at an early stage in the development cycle.

But critics of generators such as Tocher (see O'Keefe, 1984) argued that generators simply handle the parts of the model that are easy to code such as queue manipulation, leaving the more difficult part to the user. Flitman (1986) concluded that *"although a bold move toward the automated programming of simulations, providing a package which does the work for you (though not necessarily in the same way you would) can only be the start of a research drive in that direction"*.

Recently, these tools have reappeared in the literature under the description of Interactive Simulation Program Generators (ISPG). See Balmer & Paul (1986).

(b) Interactive Menu Driven Interpreter

A more modern approach to program generators was put forward by O'Keefe (1984). This Interactive Menu Driven Interpreter allows rapid development and immediate execution of visual interactive simulation models. The model can be tested while it is being written and hence the distinction between building and interaction disappears. The interpreter is programmed using a package of PASCAL simulation routines.

(iii) Visual Simulation

O'Keefe (1984) noted four ways by which a simulation model could be enhanced and which P.C. Bell (1985) divided into two categories :

Representational Graphics

1. providing a trace : values of selected variables are displayed over time.
2. displaying time series or histograms
3. displaying tables of data which are updated by the simulation (Brown, 1978).

Ionic Graphics

4. displaying a picture corresponding to the real life system being modelled in ordered to provide an animation of this real world. (see Palme (1977).

The advantages of visual simulation (Crookes, 1982) are :

- it aids in program verification by allowing the writer to determine whether the program he has written is the program he thought he had written.
- it aids in program validation because it helps the user to believe the computer program to be a fair representation of his real world.

(iv) Visual Interactive Simulation (VIS)

Hurrion conceived the term VIS. His PhD, in 1976, led to the development of VISION (Hurrion 1980), which in turn led to the Fortran based SEE-WHY system. The basic design criteria of VIS are:

1. a simulation language in which it is possible to write complex industrial problems. VIS does not constrain the user to a particular modelling approach : he may choose between event, activity or cellular-based structure (Hurrion, 1980).
2. the ability of a 1-1 correspondence between elements in the model and elements as described visually on visual display units (VDU).
3. flexibility at run time. This is the possibility to interact :
   - basic interactions which allow inspection,
   - structural interactions which allow changes.

This, in effect, permits the user to see what is happening. *'The visual interactive approach has ceased in making simulation a 'black box' technique, but now opens up the method for management to look inside. It now becomes a transparent box which greatly assists in the problem of communication and model credibility* (Hurrion, 1980)" This has made validation so much easier, that Rietz (1983) claims that it has become indispensable.

Indeed by watching a simulation model progress through time, and having the ability to interact with it, a user can improve his analysis and understanding of the original problem situation (Hurrion and Secker, 1978). This fact was highlighted by P.C. Bell (1985).

The technique of VIS was extended by Hurrion and his research students at Warwick University (Secker, 1977; Brown, 1978; Rubens, 1979; Withers, 1981; Moreira da Silva, 1982; Flitman, 1986) during a series of both internal and joint projects. The main conclusions of this work was :

1. The visual aspect has a wide appeal (Brown, 1978).

2. Interaction with the model increases confidence in it. Users feel that they are *"participants rather than spectators"* (Brown, 1978).

3. *"Situations may arise that the decision-maker may never have envisaged"* (Brown, 1978). The picture captures this. Such extreme situations can be lost in the aggregate output from a batch simulation.

4. *"There is a need for development of good interfaces that enable the decision maker to use his creative thinking and pattern-recognition capacities to their maximum potential"* (Moreira da Silva, 1982).

Other users of VIS have reported similar findings (see Crookes 1982, Ranyard and Blewitt, 1983).

A traditional problem with simulation is that it has not been possible to change underlying logic of the simulation whilst it is running. For example, in coding logic concerning movement of entities about a simulation, the analyst is setting this logic within the procedural high-level language. In an academic environment, Flitman (1986) addressed this problem by separating the logical aspect form the execution aspect. A second approach to this problem has been presented by Istel (1987) who has produced a package called WITNESS that is data driven.

2.2.6 Decision Problems within Simulation

So far in this chapter, the numerous aspects of simulation have been presented. Associated with each facet of simulation, there have been a number of complex issues that need resolving by the OR analyst before the study may continue. The first issue is whether or not to use simulation.

If simulation is accepted as the problem solving tool to be used, what type of simulation model is appropriate is the next step to be sorted out: physical models, business game models, continuous simulation or discrete simulation models ? Assuming that discrete event simulation is appropriate, a decision must be made about the hardware or software to use: mainframe or microcomputer, a high level language or some enhancement (a special purpose language or a generator, perhaps). Another decision is required about which modelling technique: three-phase, event, activity-scanning, process-interaction.

Even when the environment has been decided upon and the OR analyst prepares to proceed through the different phases of formulation, programming, verification and validation, experimental design & analysis, and implementation, there are continually decisions to be made about how to perform tasks successfully. These tasks are never trivial and the crucial parts rarely the same. Where there is repetition, tools have been devised (eg. generators and features of purpose simulation languages).

Before the formulation stage, the analyst is confronted with the need to decide how to extract the knowledge from his client. How to formulate the models, what to include, what not to include.

During the programming stage even if a generator is used there are decisions about what should be specially built.

The verification and validation phases may be made easier by visual enhancements, but the OR analyst has to make decisions about how to use these added features effectively.

Whilst designing and performing experimentation and analysis, decisions are made about how steady states may be achieved or significant results achieved or significant results collected. How, what and when to monitor are crucial aspects.

For implementation of results, the analyst has decisions to make about the level of his involvement.

For each of these problems, the user may at the top level need to decide on a strategy. Experience and expertise tell him which are better strategies. Support for this type of decision-

making may be parallelled to the need for support in other decision-making areas that prompted new computing techniques that now enable computers to perform tasks that were traditionally thought of as needing human expertise. The next section is devoted to understanding the development of tools to support decision-making.

2.3 Decision-Making and Decision-Support Tools

2.3.1 Introduction

Decision-making was described by Simon (1960) as a three-phase process :

- finding occasions for making a decision,
- possible courses of action,
- choosing among courses of action.

In effect, in the previous section on the consideration of decision-problems within simulation, the first two steps of the decision-making process were identified in the context of performing a simulation study. Indeed, the occasions when decisions have to be made were highlighted and the whole of section 2.2 has served to point to different possible courses of action. It is because the OR tools do not address these questions in their entirety that support is sought after in another discipline, namely decision-support. Firstly, a conceptual look is taken at decisions. Secondly, the idea of a decision-support system is considered. Thirdly, heuristics are considered to be of prime importance in decision-support because they offer a means of overcoming the problems of decision-making in areas where problems are not necessarily right or wrong.

2.3.2 Classification of Decisions

Simon classified decisions as "Programmed" and "non-Programmed". Lee & Hurst (1983) proposed the following criteria for the problems to which they apply :

- Programmed : " *decisions are programmed to the extent that they are repetitive and routine*".

These apply when :

1. the problem can be described quantitatively;
2. objectives and goals are well defined;
3. an algorithm can solve the problem.

- Non-programmed : *" decisions are non-programmed to the extent that they are novel...there is no clear cut and dried method for handling the problem"*. These apply when :

1. the problem is described qualitatively;
2. objectives and goals are not well-defined;
3. there is no algorithm for solution.

Keen and Scott-Morton (1978) used the classification of "structured" and "unstructured". They added to the classification the concept of "semi-structured" for problems that lay in between. A semi-structured task is one where at least one of the activities is unstructured. Characterisation of semi-structured tasks was given by Radzikowski (1983) :

"- solution objectives are ambiguous, numerous and not operational;

- the process required to achieve an acceptable solution cannot be specified in advance;

- it is difficult to determine, either in advance or after the fact, which steps are directly relevent to the quality of the decision."

And these tasks apply (Lee & Hurst, 1983) when :

1. the problem is described both quantitatively and qualitatively;
2. some goals are not well defined;
3. there can be an algorithm for the quantitative part of the problem, however it does not cover the whole problem. Therefore human judgement is an essential part of the solution process.

Each of the problems mentioned above as simulation decision problems can be classified as semi-structured.

2.3.3 Decision-Support Systems (DSS)

(i) Philosophy of DSS

It is the semi-structured tasks that can usefully benefit from Decision Support Systems (DSS) (Sprague and Carlson, 1982). Kuo & Konsynski (1984) stated that *"efficiency of decision-making can be improved by introducing new technology"* and Keen & Scott Morton (1978) stated that *"decision support implies the use of computers to assist managers in their decision process in semi-structured tasks"*. *"A DSS could be defined as a system for use in semi-structured tasks to improve the decision-making process of individuals or small groups by direct or close-intermediary contact with interactive computer systems which usually involve models* (Stevens, 1982)".

Indeed, the idea behind a DSS is that it should *"provide a coherent strategy for going beyond the traditional use of computers in structured situations, while avoiding ineffectual efforts to automate inherently unstructured ones "* (Keen and Scott-Morton, 1978).

(ii) Applications of DSS

A taxonomy of DSS was put forward by Alter in 1980 for the functions they perform :

- file drawer systems which allow direct access to data files;
- data-analysis systems which allow manipulation of data by specific and general operators;
- accounting models calculating the consequences of specific actions;
- optimisation models providing guidelines for action by generating the optimal solution consistent with a series of constraints.

(iii) Appropriateness of DSS

DSS are most useful in situations presenting the following characteristics (Keen and Scott-Morton, 1978) :

"1. The existence of a large database, so large that a manager has difficulty assessing and making conceptual use of it.

2. The necessity of manipulation or computation in the process of arriving at a solution.

3. The existence of some time-pressure, either for the final answer or for the process by which the decision is reached.

4. The necessity of judgement, either to recognise or decide what constitutes the problem, or to create alternatives, or choose a solution. The judgement may define the nature of the variable, that are considered or the values that are put on the known variables."

(iv) DSS Generators for OR

Moreira da Silva (1982) attempted to develop a DSS generator. His objectives were (see also Hurrion & Moreira da Silva, 1980) :

- to extend the application of visual modelling to areas where discrete simulation is not appropriate;
- to investigate the "potential for a generalised framework that could form the basis for development of decision making aids for different problems.

His conclusions were that DSS can be used on semi-structured problems, but current OR is not suited for semi-structured problems. This has been recognised by Eden & al (1986) where they have carried out some work on mixed strategies.

(v) Limitation of Current DSS

Radzikowski (1983) pointed out some shortcomings of current DSS.

1. Most existing information systems have attacked problems of a structured nature, whereas the most important problems are in fact unstructured.
2. *"When we consider that every problem when faced for the first time appears to us as*

unstructured, it is evident that a substantial gap exists in the scope of decision situations covered by DSS."

3. There is a discrepancy between managerial decision processes and OR solution procedures. Mintzberg (1973) points out that managers seldom make decisions as part of a deliberate, coherent and continuous decision making process. The manager (on average) spends less than 5 minutes a day on any single activity.

4. It is standard practice to interface the decision maker with a DSS through assistants rather than directly.

2.3.4 Heuristics

"Experience has shown that it is generally not possible to find optimal solutions to real-world operational problems. It is generally recognised that it is possible to systematically improve decision-making without finding optimal solutions" (Haessler, 1983). March & Simon (1958) observed that *"human problem solvers are satisficers. They establish a set of goals and then attempt to find a solution that meets or exceeds the goal set". "One important way to do this is with heuristic problem-solving procedures."* (Haessler, 1983).

A heuristic aims at studying the methods and rules of discovery, or in assisting in problem-solving. To practitioners, heuristics are simple procedures, often guided by common sense, that are meant to provide good but not necessarily optimal solutions to problems.

Lenat (1982a) hypothesized that the underlying source of power is based on the assumption that if heuristic H was (or would have been) useful in situation S, then it is likely that heuristics similar to H will be useful in situations similar to S. From this, Lenat considers that domains where heuristics may be appropriate must provide

- observability : if data cannot be found then heuristics cannot be formed and evaluated;

- continuity : if the environment changes abruptly, the heuristics may never be valid;

- stability : if the changes are continuous but rapid, the heuristic may have too short a lifetime before becoming ineffective.

By examining two case studies on the AM and EURISKO expert systems, Lenat (1982b) has drawn some tentative hypotheses about how heuristics originate :

i. heuristics can be thought of as compiled hindsight, and draw their power from the various sources of regularity and continuity in the world.

ii. heuristics rise through analogy, specification and generalisation of existing ones.

A comparison of three decision-strategies was carried out by Kleinmuntz & Kleinmuntz (1981). The decision-strategies varied in complexity :

1. random trial and error;

2. use of heuristics;

3. statistical Expected Utility (EU) maximiser using Bayes' theorem.

After trials of programs adopting these three strategies, EU was found to give the best decisions (specified by a human expert) closely followed by the heuristic strategy. The random strategy was inferior but to a lesser extent than one might have expected. Another important factor was computational time : EU was some 80 times slower than the other strategies. Haessler (1983) pointed out that a heuristic problem-solving approach generates solutions very quickly.

Kleinmuntz & Kleinmuntz (1981) thought that the results indicated the need for a trade-off between effort and decision quality, with the results indicating diminishing returns to performance from increased amount of computation effort. In particular, in situations where a good decision-rule is lacking, essentially random trial and error may be effective in discovering new environments.

A further point to note is the amount of knowledge needed about a task. EU's strategy required far more detailed information than the others, making it less adaptable to a changing environment.

Finally Kleinmuntz & Kleinmuntz (1981) point out that *"heuristic strategy seems to be*

within human capabilities, indicating that humans should be able to attain comparable performance."

Zanakis & Evans (1981) list several instances where the use of heuristics is desirable and advantageous:

i) only inexact or limited data is available,

ii) a simplified model is used;

iii) an exact method is not available;

vi) there is a repeated need to solve the same problem frequently or on a real time basis;

vii) heuristic solution may be good enough for a manager if it produces results better than those currently realised;

ix) as a learning device;

x) there are other resource limitations (eg budget, manpower).

Zanakis and Evans (1981) have proposed that the criteria for a good heuristic are simplicity, speed, accuracy, robustness, good stopping mechanisms that take advantage of search 'learning ' and avoid stagnation. Haessler (1983) disagreed with the point on simplicity. He argues that a *"rational user should be concerned with the quality of the answer and the effort to get to it, not how easy it is to describe the procedure."* It is also pointed out that problem-specific heuristics will tend to be more efficient than general ones, but will also be less flexible (Zanakis & Evans, 1981). This may suggest the advantages of a system which modifies the basic heuristic according to the specific problem.

Thorngate (1980) attempted to compare ten general purpose heuristics by trying to determine how often each would select alternatives with highest-through-lowest expected value in a series of randomly generated decision situations. His results indicated that most of the heuristics, including some which 'ignored' probability information, regularly selected alternatives with highest expected value, and almost never selected alternatives with lowest expected value. Such results tend to imply (together with Kleinmuntz & Kleinmuntz, 1981) that overheads involved in

developing and running elaborate 'accurate' heuristics, far outweigh the advantages gained.

However, a problem with heuristics is that they do not state the assumptions under which they hold (Rajagopalan, 1986).

2.3.5 Considerations about DSS and Simulation Decision-Problems

It has not been the objective of this section on decision-making and DSS to provide a detailed study on the subject, but rather to parallel the decision problems of simulation with the decision-areas under investigation in the field of DSS. The identification of simulation decision-problems as semi-structured makes it conceptually appropriate and attractive to consider decision-support tools for simulation decision-problems. However, it also appears that more powerful decision tools would be required, more in the line of the ideal DSS postulated by Benet (1983). This ideal DSS would take an active role in leading the decision-maker to a decision. To achieve this, the computer must have :

- an understanding of what the decision-maker is trying to do;
- an understanding of the decision-making environment;
- a knowledge based system.

The concept of "understanding" is associated with "intelligence". "Computer understanding" relates to the domain of "artificial intelligence". In the next section, an investigation will be carried out on the potential of artificial intelligence in simulation decision-problems.

2.4 Decision-Making and Artificial Intelligence (AI)

2.4.1 Introduction

Within computer science, AI has perhaps the greatest potential with respect to decision-making (Keen and Scott-Morton, 1978). If one can understand computer intelligence, one is well on the way to understanding human intelligence, and therefore human decision-making (Winston, 1977, Young, 1979).

Four core topics of AI as listed by Shortliffe (1976) are :

- modelling and representing knowledge;
- reasoning, deduction and problem solving;
- heuristic search;
- AI systems and languages.

The first three are directly relevant to decision-support (Keen and Scott-Morton, 1978). The fourth deals with the most applicable area of AI namely Expert Systems (ES). Keen and Scott-Morton regard ES as one area of AI especially relevant to decision-support. Radzikowski (1983) extrapolated and conceived the Decision Support Expert Systems (DSES).

"The DSES would :

- *operate on explicit knowledge stored in a knowledge base;*
- *interface with the decision maker in a natural language.*
- *choose proper quantitative methods (QM) techniques;*
- *create a qualitative model of the decision situation;*
- *present and explain the solution to the decision maker."*

In this section on decision making and AI, the focus is on applicable AI techniques and in particular expert systems.

2.4.2 Artificial Intelligence : Expert Systems

Artificial Intelligence is concerned with designing computer systems that, for certain areas, emulate some of the characteristics of human thought : the ability to learn, reason, solve problems, and understand ordinary human languages (Shannon 1986). *"An AI technique is a method that exploits knowledge that should be presented in such a way that :*

- *it captures generalisations;*
- *it can be understood by people who must provide it,*
- *it can be easily modified;*
- *it can be used to correct errors and to reflect changes in the world and our world view;*
- *it can be used in many situations even if it is not totally accurate or complete;*
- *it can be used to help overcome its own sheer bulk by helping to narrow the range of possibilities that must normally be considered."* (Rich 1983).

Research and applications into how humans acquire, organise and use knowledge fall into two broad categories :

1. those which attempt to duplicate or imitate the natural human abilities (vision, language processing, etc) in the same manner as human beings perform them.
2. those which attempt to duplicate the results of learned skills or expertise without concern for whether it is exactly the process used by the real expert. There are some purists, though, who will argue that this has very little to do with Artificial Intelligence (see Cole, 1986).

This second class of applications include "expert systems". Expert systems are concerned with the automation of tasks that are normally performed by specially trained or talented people. Expert Systems or knowledge based systems are designed to compile the experience of any number of experts in a given field into a series of rules; rules which are then used to draw inferences and suggest to the user (or carry out automatically) a course of action to deal with a given problem. According to Buchanan (1982), ES differ from other AI programs in respect to their utility , performance and transparency.

1. Utility : *"designers of expert systems are motivated to build useful tools in addition to constructing programs that serve as vehicle for AI research"*.
2. Performance : *"the hallmark of expert systems is high performance"*.
3. Transparency : *"it is not necessary that expert systems be psychological models of the reasoning of experts. However, they must be understandable to persons familiar with the problem."*

(i) Some Examples of Expert Systems

The mass-spectrogram interpreter DENDRAL (Feigenbaum, 1971) was the prototype of all expert systems, but the most influential must be considered to be MYCIN (Shortliffe, 1976). MYCIN is a computer system which diagnoses bacterial infections of the blood, and prescribes suitable drug therapy. It has inspired a whole series of medical-diagnostic clones (eg PUFF (Kunz & al, 1978)).

Expert systems received much publicity after a couple of important success stories. In the 1970's, a group of researchers at SRI International developed PROSPECTOR, a rule-based expert consultation system for examining data about sites likely to have valuable bodies of ore, such as porphyry copper deposits. In 1981, PROSPECTOR accurately predicted the location of an important molybdenum deposit in the area of Mt. Tolman in eastern Washington State, the value of which is in millions of US dollars (Gashnig, 1982).

One of the first expert systems to be used in manufacturing was R1 (later renamed XCON), which was developed at Carnegie-Mellon University under contract to Digital Equipment Corporation. The program was applied to the task of configuring VAX computers. Unofficial estimates of the savings to DEC achieved by the use of this program range from 10 to 20 million US dollars annually (Walker & Miller, 1986).

There are many other examples of profitable expert systems (see Forsyth, 1984).

(ii) Architecture of Expert Systems

Expert Systems differ from traditional programs (and therefore from conventionally understood DSS (Stoner, 1985)) in both their architecture and in the development process by which they are constructed. In contrast to traditional programs where the emphasis is on procedural instruction for the computer, the focus of expert system development is on knowledge acquisition and organisation of knowledge bases (knowledge engineering). *"The power of an expert system derives from the knowledge it possesses, not from the particular formalisms and inferences it employs* (Feigenbaum and Mc Corduck, 1983)". And Davis (1977) argues that *"high performance requires large amounts of knowledge"*. To construct and maintain such large knowledge bases, it is necessary to separate domain-specific knowledge from problem solving methods (Buchanan, 1982). In a typical ES, knowledge is organised into three categories (Gevarter 1982; Yadzani 1984) :

- a global database or blackboard which provides the input data (sometimes defined as declarative knowledge) defining the particular problem and keeps track of the current solution status or solution;
- a knowledge base or rule base which describes the facts and heuristics associated with the general problem domain. Since this domain knowledge often takes the form of rules, it is sometimes referred to as procedural or relational knowledge;
- a control or inference structure which defines the problem solving approach or how the data and knowledge can be manipulated to solve the problem. This is usually known as the inference engine.

It must stressed that other names are found in the literature that might even appear as conflicting with the above categories. Harrison (1984) uses the following corresponding classification factual, judgemental and procedural. One of the key issues in the construction of an ES is the problem of how to represent and store the necessary data at the appropriate levels.

(a) knowledge representation in the Data-Base at Data level

There are several ways to represent declarative knowledge :

- predicate calculus (logic) has been found particularly useful in AI because through it, problems can be adequately represented symbolically. The real strength of predicate logic is that ordinary English sentences can be recast into a formal representation that can be handled by a computer and compared to other information.
- frames are data structures in which all knowledge about a particular object or event is stored together. A frame is a collection of facts and data about something or some concepts. Most frame-based knowledge representation schemes include the idea of having different types of frames for different types of objects with fields or slots in each frame. The disadvantage of frames is that they are less modular than other representations, making changes possibly more complicated (Adelsberger, 1986b).
- semantic network : declarative knowledge can also be represented in a graphical representation. Semantic nets are like frames in the sense that knowledge is organised around the objects being described, but here the objects are represented by nodes in a graph and the relations among them are represented by labelled arcs. (Brachman, 1979). Semantic networks permit the inheritance of information. The disadvantage of this representation is that it is difficult to deal with exceptions : inheritance of multiple sources can cause problems (Adelsberger, 1986b).

Other representations exist (eg scripts), but as Buchanan (1982) points out : *"Almost any knowledge can be represented in almost any formalism; the main issue is how easily the domain knowledge can be codified and maintained"*.

(b) knowledge representation in the Rule-Base at Domain Knowledge level

This knowledge level describes the problem or domain knowledge specific to the particular kind of problem that the system is set up to solve (ie manufacturing systems, distribution systems, computer networks, etc.) This knowledge is usually procedural (or relational) in the sense

that it tells how the data for a problem can (or should) be manipulated in order to solve the problem. Power in an expert system lies in its specific knowledge of the problem domain. Most expert systems contain hundreds of rules obtained by seeking out the knowledge possessed by the best experts and then linking this knowledge to form rule networks. All of the techniques identified in the data level section are also applicable to the representation of domain knowledge. Additional ways exist to represent this knowledge :

1. Conventional computer program : if the solution procedure is well understood and can be programmed as an algorithm, then a conventional program can be written to manipulate the data to get a solution.
2. Pattern-invoked program : here, the domain dependent knowledge is encoded in the form of operators or pattern invoked programs which are activated by control structure when certain conditions hold in the data.
3. Production rules : one type of pattern evoked programs of particular interest is the production rule. These are programs of the form :

   "IF <condition> THEN <primitive action>"
4. Logical representation : using predicate logic. A solution is found by doing a tree search until a set of assertions is found that provides a solution.
5. Conditional probabilities : knowledge is also sometimes represented in terms of conditional probabilities that various events will occur given that other events have occurred.

(c) knowledge representation in the Inference Engine at Control level

At the control level, the computer program makes decisions about what is the question to be answered and then the control strategy of how to use the database and knowledge base to solve the problem at hand. Control strategies can be encoded as meta-rules (Davis & al, 1977; Davis, 1980a). and can be classified as either irrevocable or tentative. In an irrevocable control strategy, an applicable rule is selected and applied irrevocably with no provision for reconsideration later.

In a tentative control strategy, a rule is selected, the rule applied, but provision is made to return to this point of computer later to apply a different rule if no satisfactory solution (or no solution) is found.

The control strategy is a function of the problem to be solved and several approaches can be used including :

1. Forward chaining : if the solution starts from an initial set of data and conditions (the existing global database) and moves towards some goal or conclusion, it is called model-, data-, event- or antecedent-driven.

2. Backward chaining : if the desired conclusion or goal state is already known, but the path to that conclusion is not known, then working backward is called for and the model is said to be goal- or expectation-driven.

3. Problem reduction : in this technique, the problem to be solved is partitioned or decomposed into subproblems that can be solved separately. This approach which yields problems with a smaller search space is applicable for problems in which a number of non-interacting tasks have to be done to achieve a goal. DENDRAL (Feigenbaum & al, 1971) is an example of a forward chaining model using problem reduction, while MYCIN (Davis & al, 1977) uses backward chaining and problem reduction.

4. Constraint propagation : in this problem solving technique, the set of possible solutions becomes further and further constrained by rules or operators that produce "local constraints" on what small pieces of the solution must look like. Constraints are viewed as relationships (or subgoals) that must be satisfied (Stallman and Sussman, 1977; Stefik & al, 1982).

(iii) Implications of importance of knowledge

Because acquiring the knowledge for an expert system is so crucial, a number of themes have evolved from this process. These are automatic knowledge acquisition and learning systems, the role of heuristics, and the issues of dealing with incomplete knowledge.

a. automatic knowledge acquisition & learning systems

From Buchanan & al (1983), it appears that one of the major difficulties in knowledge acquisition is the mismatch between the way a human expert normally states knowledge and the way it must be represented in the program. To assist in this phase, a number of aids have been developed which Buchanan & al (1983) see as *"an attempt to shift the responsibility for parts of the knowledge acquisition process from human to computer"*.

In this context, learning systems have appeared (Rada, 1984). (Smith, 1984). Forsyth (1984) defines machine learning as any automatic improvement of a computer system over time as a result of experience. Examples are BEAGLE, EURISKO, TEIREISIAS , EMYCIN (see Politakis and Weiss, 1984)

Laurie (1985) describes the concept of "Intelligent Database", which would *"run around a database full of facts and extract from the data a set of rules. These rules would be expressed either as statements in a human like language which would convey the substance of them to a person so that he can make the appropriate decisions on new data; or as programs which will allow the computer to make the same decisions automatically"*. Laurie describes a system called Analogue Computer Learning System (ACLS).

Complete environments have been developed to assist with the knowledge engineering process. Jones (1984) describes a system called REVEAL. *"One of the primary objectives of REVEAL was to bridge the "gulf between knowledge engineering and the main stream of operations research and management science activities"*.

b. dealing with incompleteness of knowledge

The problems of knowledge that is incomplete can be dealt with in a number of ways. Quinlan (1983) talks about INFERNO which is developed to make inferences when knowledge is vague, uncertain or probabilistic. Its major contributions are :

- the guaranteed validity of any inferences;

- its concern for, and assistance, in establishing the consistency of the information about the problem and its domain.

Doyle (1983) produced a paper on the case of judgements and reasoned assumptions.

Cohen and Grinberg (1983) provide a good review about the current approaches to uncertainty. They propose a theory of heuristic reasoning about uncertainty.

c. Role of Heuristics

Builders of expert systems attribute the performance of their program to the body of knowledge they embody, which includes a large number of judgemental rules (heuristics) which guide the system towards plausible paths. The operation of the expert system through its rule base to finding appropriate actions has needed some means of constraining the search and some authors have resolved to heuristics (see Georgeff, 1983).

(iv). Modes of Use of an Expert System

Michie (1980) observed that there are three different user modes for an expert system :

1. improving or increasing the system's knowledge : user as a tutor;

2. getting answers to problems : user as a client;

3. harvesting the knowledge base for human use : user as a pupil. This is stressed by the role of explanation (Davis and Lenat, 1982)

(v). Appropriateness of Use of Expert Systems

Expert systems are limited to handling decisions that take humans anywhere from a few minutes to a few hours to make. Decisions taking less than 45 seconds are evidently not very critical; otherwise, the expert would presumably take more time with them (see Johnston, 1983). At the other end of the spectrum *"you could spend hours watching your expert waiting for that*

infrequent situation when a special rule applies ". Problems requiring extensive analysis are also inapplicable : " *you probably won't be able to develop the rules before the rules or environment change*".

The main finding in the "Report to the Alvey Directorate on a short survey of Expert Systems in the UK Business" (d'Agapeyeff, 1984) was that : "*simpler expert systems are practical and are being implemented now by self taught teams with little risk at relatively low cost to produce modest but unusual gains*".

From this it was concluded that "*it is necessary to correct the impression that expert systems are inherently complex, risky and demanding. This impression is a handicap to competitive developments in the supply and usage of Advanced Information Technology*".

There is a common view that the difficult task of plausible reasoning from uncertain knowledge is fundamental. Yet during the survey carried out for the Alvey report, no team questioned about the treatment of knowledge said that uncertainty was a fundamental existing feature although it was sometimes anticipated to be of growing importance in future applications. Even then, it was not certain that the uncertainty could not be handled by an ad hoc solution.

There is considerable evidence that individuals do not often use probability information as they should (Tversky & Kahnerman, 1974). This failing has almost always been attributed to cognitive limitations associated with predecision activities. Thorngate (1980) proposes two alternative explanations:

1. "*Individuals may ignore or misuse probability information because they may simply not notice, or not care, about small or infrequent decrements in reward which result from this ignorance or misuse*".
2. "*Alternatively, they may ignore or misuse probability information because the time or effort required to use it properly may be more costly than any decrease of pay-offs associated with their occasional suboptimal choices*".

Buchanan (1982) summarised the state of the art :

- narrow domain of expertise;
- limited language for expressing facts and relations;
- limited assumptions about problems and solution methods;
- stylised line of reasoning;
- little knowledge of own scope and limitations;
- knowledge base extensible but little help available for initial design decisions;
- single expert as knowledge Czar.

(vi) Problems with Expert Systems

There are a number of problems associated with expert systems (M.Z. Bell, 1985).

- expert is not available;
- expert is unable to communicate his ideas;
- expert is unwilling to communicate his ideas;
- there is no expert.

There can be problems with the knowledge-representation language : it may be too rich, not rich enough or incompatible with the experts knowledge.

Finally testing the system may be a problem due to the fact that :

- the system is being designed to cope with extraordinary events;
- the systems covers too wide an area;
- it is too expensive to test the system;
- it will take too long to test the system;
- what constitutes correct performance may not be known.

Martins (1984) offers a sceptical point of view suggesting that the success stories in expert systems were due to brilliant programmers, easy problems, lot's of time and generous funding, the fact that commercial tools were not used, and luck. He also notes that programs do not often work as advertised.

Stoner (1985) warns against the fact that there may be some financial advantage of calling DSS expert systems because of their status.

Adelsberger (1986b) points to some of the weaknesses of expert systems :

- many AI programs require large amounts of computing power;
- large amounts of knowledge are required;
- many parts of intelligence are poorly understood.

(vii) Acceptance of Expert Systems

It seems likely that the range and utilisation of expert systems may depend on the provision of :

- good dialogue facilities (Hebditch 1984); acceptance will be dependent on the system's naturalness (Davis, 1977).
- the capacity to embed an expert system within a conventionally designed application;
- the sharing of state tables and other control information between expert systems and other programs.

(viii) Expert System Languages and Shells

There are two possible decisions regarding the building and implementation of an expert system :

(1) - write a program from scratch using a general AI language (such as LISP or PROLOG) or an object-programming language (eg. SMALLTALK);

(2) - use a specific expert system "language" or shell with a built-in inference mechanism (eg.

SAVOIR, ROSIE, EMYCIN, META-DENDRAL).

The first possibility requires a great deal of time and effort since the analyst must first decide how he is going to represent the knowledge contained in the expert system, and then express the knowledge in that form (ISI, 1984).

A shell on the other hand provides the basic building blocks, but does have the big restriction of enforcing knowledge to be represented in a specific way. Shells have been designed to make expert systems easier to write. In use they take the form of a special programming language with a built-in inference mechanism (ISI, 1984). Some powerful shells are SAVOIR (ISI, 1984), CRYSTAL (Birnie, 1987).

In a destructive article about expert systems in general, Martins (1984) argued that *"for the most part, current off-the-shelf expert system tools (shells) cost too much, are hard to use, yield very inefficient programs, and seem to have sharply limited applicability to complex real world problems"*. In line with this, Gashnig (1982) argues that his work on PROSPECTOR illustrates that ES intended for actual practical use must accommodate the special characteristics of the domain of expertise : therefore development from a language is more appropriate.

By 1980, of the expert systems currently in use, the dominant method of implementation was to use INTERLISP on a PDP-10 (Bond, 1981). This is a modified version of LISP, containing facilities for explanation, rule-acquisition and debugging. Prolog is beginning to get wider acceptance as being suitable for expert systems implementation, especially since its syntax closely resembles that of production rules, which are the accepted way of representing a large amount of knowledge in an expert system. Many modern expert systems are now written in Prolog. The importance of Prolog relates to its role in the fifth generation computer systems project.

2.4.3 Future AI : the Fifth Generation

(i) Background

Current conventional computers have become numerical-processing oriented, stored-program sequential processing systems. However, the situation has evolved in the following ways (Treleaven & al, 1982; Lemmons, 1983):

1. VLSIs (Very Large Scale Integrations) have substantially reduced hardware costs;
2. A new architecture for parallel processing is now required because device speed has approached the limit for sequential processing;
3. Parallel processing should be realised in order to utilise effective mass production of VLSIs;
4. The current computer technology lacks the basic functions for non-numeric processing of speech, text, graphics and patterns, and for AI fields such as inferences, association and learning.

(ii) The Japanese Fifth Generation Computer Systems Project

It is for these reasons stated above that the Japanese embarked upon the Fifth Generation Computer Systems (FGCS) project. The functions required of such a system are as follows (Brooking, 1984):

- Problem-Solving and Inference Function
- Knowledge-Base Function : this is aimed at providing storage and retrieval of not only data but also reasonable judgements and test results organised into a knowledge base;
- Intelligent Interface Function : this is intended to allow computers to handle speech, graphics and images so that the computers can interact with humans flexibility and smoothly.
- Intelligent Programming Function : the ultimate goal is to allow computers to take over the burden of programming from humans.

(III) Prolog

" Research in the initial stage of the FGCS project is based on new programming languages, the version 0 kernel language, which is extended on Prolog " (Institute 1983).

Prolog is a programming language based on Kowalski's procedural interpretation of Horn clause predicate logic (Kowalski,1974, 1979). The language was developed and first implemented by Colmerauer's research group in Marseilles (Colmerauer & al, 1973; Roussel, 1975). As argued by Clark & McCabe (1982), Prolog provides much of the inference machinery that the expert system implements has to program up in LISP. (Eg.Build in backtracking device).

Kowalski claimed that logic programming was the universal solution to any problem. A statement described as dangerous by Lehman. Feigenbaum also asserted that it would be "foolish to switch over to Prolog when LISP is available and proved" (interview with Johnson, 1985).

Why use Prolog was put forward by Pope (1985).

> *"Prolog is an implementation of Horn subset of first-order predicate logic in the form of a programming language. It is especially well suited to the design of programs for ES, natural language interfaces, pattern matching and as a rapid prototyping language. In PROLOG, the programmer does not normally work defining functions and data structures as with other languages, but rather builds hierarchies of predicates which can be likened to statements or facts or relationships between facts".*

Why not to use PROLOG was argued by Forsyth (1984) :

- Prolog has very little error-protection against mis-spelling;
- the user must understand the implementation details of the backtracking mechanism to write code;
- it relates the order of clauses to their meaning;
- built-in predicates have side effects;
- everything in database is global;
- Prolog is already riddled with non-standard 'enhancement'.

2.4.4 Conclusion

From a conceptual level, it would appear that expert systems are appropriate for the decision-problems of simulation. Knowledge of simulation experts could be pooled and put into knowledge-bases. Support for decision-making is then potentially available in the form of knowledge systems. Some guidelines proposed by Shaw (1987) are that knowledge support systems require that:

1. the tools be domain independent;
2. the tools be directly applicable by experts without intermediaries;
3. the tools be able to access diversity of knowledge sources;
4. the system should be able to present knowledge from a diversity of sources with clarity;
5. users should be able to apply knowledge in a variety of familiar domain and freely experiment with its implications;
6. as the overall knowledge-support system develops, it should converge to an integrated system.

At a practical level it will be found in the next section that some work has been carried out in this context.

2.5 Simulation and Artificial Intelligence

2.5.1 OR and Artificial Intelligence

The prime advantage of AI may be seen as offering a new methodology to tackle complex problems by making use of subjective and heuristic methods similar to those used by humans. *"This enables processing of problems in a manner which may be suboptimal but which corresponds to human levels of performance... Human reasoning has the ability to spot the essential elements in a problem and pattern in data, thus structuring the problem situation and allowing a qualitative analysis* (Phelps, 1986)". Though it may make use of heuristics, OR on the other hand often attempts to build scientific models which are rigorous and objective in nature. *"This leads to problems when dealing with complex systems"*

The natural conclusion of this argument is to link these two fields conceptually. This would mean that in the domain of a decision-situation previously amorphic, a structure can be imposed, making the problem semi-structured and thereby permitting the use of OR techniques to provide assistance (Radzikowski 1983).

Similarities between these subjects exist, both in the problems they face and in the techniques they use (Phelps 1986) :

- both OR and AI approaches build models;
- both use 'heuristic approaches' in the absence of optimal ones;
- both use mathematics;
- both use computer implementations;
- both employ interdisciplinary teams.

In using Expert Systems, several distinct approaches are possible :

1. using standard expert system methods;
2. embedding OR techniques within expert systems (O'Keefe, 1985);
3. embedding expert system techniques within OR (Flitman 1986).

Further suitability of AI to OR, maybe seen by considering the OR person and the development of Expert Systems. Indeed, the critical person behind the development of an Expert System is the knowledge engineer. He has three distinct tasks :

1. the extraction of knowledge from human experts;
2. the organisation of this knowledge as a knowledge base;
3. the use of the knowledge base in making reasoned deductions.

The first area -knowledge acquisition- has been implicitly developed within OR and practised for some time (eg Eden, Jones and Smith 1983). The methods used in the second branch -knowledge organisation- are modelling processes, and are thus familiar to the OR scientist in approach at least (O'Keefe 1985).

2.5.2 Combining Simulation and Artificial Intelligence

It was natural that when the OR community developed an interest in AI that the simulationists should be active. *"The thrust of much simulation and expert systems work is similar : to provide a computer-based model which aids decision making"* (O'Keefe 1986a). Similarities between Simulation and Expert Systems have been highlighted by a number of authors :

- Simulation and Expert Systems methods are similar in that they are each based on a modular representation of a system, with an inference mechanism that drives this representation (Oren 1986b, Vaucher 1985).
- the resemblance between the conditional event and the production rule (O'Keefe, 1985; Vaucher, 1985).

O'Keefe (1986a) has proposed a taxonomy for combining simulation and expert systems :

- embedding an expert system within a simulation model;
- embedding a simulation model within an expert system;
- parallel operation : simulations and expert systems that are designed, developed and implemented as separate software may interact;

- cooperation : the simulation and the expert system may share some data;

- intelligent front end : this is an expert system that sits between a package and the user, generates the necessary instructions or code to use the package following a dialogue with the user, and interprets and explains results from the package. Useful intelligence includes :

(1) dialogue handling (a natural language interface or at least user-directed free format input);

(2) some model of the user so that the system adjusts its requirements of the user, given evidence that the user is inexperienced, experienced or whatever.

(3) a model of the target package, so that some decision can be taken by the intelligent-front-end rather than referred to the user.

2.5.3 Current Research

The area of artificial intelligence and knowledge-based systems offers great potential for enhancing the support of simulation modelling (Shannon, Mayer and Adelsberger, 1985; Shannon, 1984). The explosion of interest in this field can be seen by the number of conference papers on this topic (Kerckhoffs, Vansteenkiste, Zeigler, 1986; Luker and Adelsberger, 1986; Luker and Birtwistle, 1987; Henson, 1988; Huntsinger, Karplus, Kerckhoffs, Vansteenkiste, 1988).

Conventional simulation languages are limited by the necessity to settle on fixed, simplistic resolutions to a number of complex tradeoff decisions (Rozenblit & Zeigler, 1985). The introduction of intelligence into such languages promises to yield unprecedented levels of modelling support, providing a means to capture complex concepts and to adapt to different situations (Ruiz-Mier & al, 1985). In this section, a review is made of the current research that is combining some aspect of artificial intelligence with simulation. The objective is to investigate how some of the decision problems of simulation have been tackled.

The following classification is considered though many projects may belong in more than one area:

- Intelligent Simulation Environment
- Integrating Simulation into Knowledge Based Systems
- The use of Prolog

(i) Intelligent Simulation Environment

Luker (1986a) wrote in his preface to the proceedings of the Conference on Intelligent Simulation Environments that the conclusion had been reached that *"from the experienced user to the novice, all have found simulation to be lacking"*. A simulation language is not enough for either extremes of the user spectrum. *The expert needs more support than a language can provide. There must be an integrated software environment to support all stages of the simulation process from requirement specification to the analysis of results.* The novice on the other hand does not have the knowledge or experience required to utilise a simulation language effectively. However simple a language is claimed to be, the inescapable fact remains that programs must be written. To get the best out of any language, some kind of knowledge of simulation techniques and algorithms is essential. The software environment must provide a simple interface while offering expertise within the system to construct the simulation for the user. Given a software support environment with expertise, the user is then free to concentrate on the problem rather than having to be concerned with simulation techniques and practices.

To overcome some draw-backs in the "classical" modelling and simulation (lack of flexibility in expressing model structures, extensive programming efforts), an alternative approach is to consider an AI knowledge representation system to express knowledge in a simulation model (knowledge based simulation). Several topics among others in this framework are the applications of various knowledge representation schemes, the development of so-called object-oriented modelling languages, natural language understanding systems and advisory systems.

a. Knowledge-Based Simulation

Reddy (1987) is considered the pioneer of knowledge based simulation (KBS). Reddy argues that a KBS can be described by the features it should possess. The ideal KBS should :

- accept a description of a problem and synthesize a simulation model by consulting an appropriate knowledge base;
- accept a goal in the form of expectation
  - select model;
  - determine performance metrics;
  - generate search space of plausible scenarios;
  - execute simulation model by controlled selection of scenarios;
  - recommend a scenario that satisfies the stated goal.
- explain the rationale as to why only certain scenarios have been explored and why it recommends a particular scenario.
- learn from experience and disclose its behaviour.
- display the resultant model built by the KBS.

Reddy argues that the key to realising these functions in KBS is the use of the same programming paradigms used by knowledge-based expert systems.

Robertson (1986) proposed a rule-based expert simulation environment using the production rule paradigm as the basis for describing and coding the processes to be simulated. The anticipated advantages are that it will be easy to build and easy to maintain.

Khoshnevis & Chen (1986, 1987) present a rule-based system that aids the user with little or no knowledge of simulation model building to construct simulation models. The system automatically creates the corresponding computer program.

Unger et al (1986a, 1986b) describe JADE, being an environment that supports the development of distributed software. Components may be written in any of a number of different

languages (such as Ada, C, LISP, Prolog, SIMULA). A common inter-process communication protocol provides a uniform interface among the components such as its supports integrating simulation and knowledge-based systems. The tools and their use in the development of distributed software are described for the control of a simulated system of parking lots.

Some other examples of KBS are FRAMEWORKS (McVicar & Smith, 1988), KBS in Control (Kornecki, 1988), the work on KBS for manufacturing by Zalevski (1988) and also for FMS by Soeterman & Bond (1988).

b. Object Oriented Programming

Object-oriented modelling languages offer a most sophisticated way of representing knowledge in a simulation model. Here the knowledge-bases contain :

- various facts about each entity in the system concerned;
- knowledge about the entity's relationship to other entities;
- knowledge about the relationship between entities and constituent specifications.

Adelsberger (1986a) summarised the philosophy of object-oriented programming:

- first the user creates objects that correspond to external world objects, and represent modular components of the real world.
- the behaviour of the simulation model's objects describe the behaviour of the external objects and how these objects will behave in response to various inputs.
- objects act upon each other by passing messages describing both functional and relational actions. Messages passed between objects are the carriers of all interactions between objects.

In essence, the object-oriented 'world' of this simulation environment consists of messages of information that provide behavioural rules (objects embedded) and manipulation specifications (message embedded). Example of object-oriented systems are Smalltalk (Goldberg & Robson, 1983) and ROSS, (Klahr, 1986).

Moreover, knowledge is processed about the effect of actions in the system. Some goal features are :

- interactivity in model creation and alteration (knowledge representations are both flexible and extendible);
- little programming effort in the creation of models;
- automatic examination of consistency and completeness.

Middleton and Zanconato (1986) describe BLOBS, an object-oriented framework, implemented in the artificial language POP11 within the POPLOG environment. The framework resulted from a study into the most promising techniques for building an expert system to model the decision-making process of a ground controller monitoring aircraft tracks on radar. BLOBS builds upon concepts from SIMULA and ROSS. The language emerged from artificial intelligence and simulation and is currently used in the area of planning. For the future, potential uses are policy formulation, training purposes and interfaced to real data advisory functions.

Klahr (1986) discusses the issue of expressibility within the object-oriented simulation system called ROSS. To test ROSS, it has been used to develop two battle simulations : TWIRL (a tactical simulation) and SWIRL (which simulates an offensive air attack of invoking bombers on a defensive force of radars, fighters and control centers).

Blakemore, Dolins and Thrift (1986) describe the design of a general purpose robotic vehicle simulator based on object-oriented simulation. Their simulator is meant for the rapid prototyping of distributed expert systems to be applied in the robotic vehicle.

Other examples of the presence of object-oriented programming are the development of IMP (Integrated Modelling Package) by McLaren & al (1988) which is an object-oriented module for manufacturing; and also the work of Witte & Grzybowski (1988) on the development of MODULA-2 which is an object-based simulation language.

c Natural Language Understanding Systems (NLUS)

Doukidis and Paul (1985, and Paul & Doukidis, 1986) describe two early attempts to automate the formulation of a discrete event simulation model. The first prototype system, written in LISP on an APPLE microcomputer, is an expert system built using the MYCIN approach. Inflexibilities in this system led to a more informal approach when constructing the second prototype. The resultant system is a natural language understanding system.

In later publication, Paul & Chew (1987) suggest that the output from the NLUS, which is written in Pascal on a VAX, can be input to an ISPG where the quantitative aspects of the problem need to be added then a simulation program generated.

d Advisory systems in modelling and simulation

The basic issue here is the intelligent computer-aided guidance of the simulation analyst through the different stages of a simulation study.

Elmaghraby and Jagannathan (1985) propose an expert system for simulationists to assist in selecting a simulation language matched to their model and their computer resources.

Bondi and Pritsker (1986) describe a procedure for automatically recognizing the admissibility of some techniques (queueing, network algorithms and a semi-Markov process) and an analysis procedure for the analysis of an IDEF2 model. IDEF2 is a definitional language to describe the functions, information and dynamic aspects of manufacturing systems. The development of NET-TRANS implements the two mentioned analysis procedures.

Lehmann & al (1986) discuss the implementation of several modelling support tools for dialog-oriented knowledge-based and problem-adapted model construction and model execution with respect to given evaluation objectives.

Fjellheim (1986) presents the KIPS project, a Knowledge based Interface to Process Simulation. In this project an intelligent front end will be developed to process flow sheet programs. The goal of the KIPS project is the design of an intelligent assistant to the modeller. By using

KIPS, the knowledge involved in the design process can be preserved for later and other uses.

Rao and Sargent (1988) have developed an experimental advisory system for operational validity (OVAS). OVAS determines whether or not the model's output behaviour has sufficient accuracy for its intended purpose or use over the domain of the model's intended application.

de Swaan Arons, Jansen and Lucas (1986) present the possibilities of DELFI-2. DELFI-2 is a tool for the building and consulting of expert systems, similar to the notion of an empty shell. Production rules, frames, etc can be used for the possibility to interact with existing databases and simulation packages.

In the line of expert system shells, Futo (1988) described ALLEX which is a CS-Prolog based system shell to support simulation.

Wales and Luker (1986) present an environment for discrete event simulation, written in ADA, that is designed to be used by expert and non-expert simulationists. A discrete event simulation program is generated on request therefore reducing the need to learn a specialist programming language.

Flitman & Hurrion (1987) discussed the merits of linking discrete event simulation models with expert systems in order to

- control the simulation,
- monitor the simulation,
- actually contain parts of the simulation logic.

Hill & Roberts (1987) developed a knowledge-based advisory system at model development stage. Their work focussed on the design and implementation of a system which could support their simulation students in debugging their models.

(ii) Integrating Simulation into Knowledge-Based Systems

The focus of interest lies in simulations with multiple decision-making agents, each performing tasks for which AI is well suited (such as planning, scheduling, hypothesis formulation,

sometimes performed in an environment full of uncertainty and incomplete and distorted information). For such systems, those subprocesses which are clearly heuristic and symbolic (eg decision-making) could be modelled by expert systems while the remaining parts (eg physical processes) are conventionally modelled simulation systems.

Evaluating knowledge-based control systems can be achieved by linking to a real world process. However, this might be impossible or undesired in many cases. Therefore testing these systems should be done by linking them to a simulation model of the process. It provides the tools to test the system thoroughly and to estimate its capabilities and expected performances (eg. Blakemore, Dolins and Thrift 1986).

Another example of applying simulation to test knowledge-based systems is in the design of particular architectures for knowledge-based systems. For instance, parallel knowledge-based systems concepts are being studied, where a number of rule processors are simultaneously operating on a common rule-base and (multi-accessible) database. Simulations of a parallel design are applied on a sequential computer to study the system's performance dependent on some parameters (eg Groen & al, 1986).

Fujiwara and Sakaguchi (1986) discuss the practical use of expert systems in power system planning. The expert system presented is composed of a primary interactive analytical system to perform load-flow computations and a knowledge-based system. A planner can be guided by the load-flow engine in the expert system to computer scenario's and explain suggestions. In this way, a platform is created to increase creativity and to allow certain novice planning as well.

(iii) Use of PROLOG in current research

In this section, the interest lies in the consideration of the popularity of Prolog to address a diversity of simulation issues.

Futo, Gergely and Deutsch (1986) give a description of modelling as a cognitive process and the key role of Systems Theory in this process of model construction. A modelling and problem-solving system called TS-Prolog which is based upon a restricted version of the first

order calculus, is considered and its practical use is demonstrated on the basis of an insulin administration problem.

Vaucher & Lapalme (1986) considered process-oriented simulation in Prolog and presented SIMPOOPS. Along the same line, Doman (1988) presented OBJECT-Prolog which is a dynamic object-oriented version of Prolog. Adelsberger (1984) investigated the suitability of Prolog as a simulation language. Flitman developed simulation models using Prolog (1986). Radiya & Sargent (1986) also looked at logic programming with discrete event simulation and concluded that *"the widely used LP processors like Prolog and LogLisp are not directly suitable for simulation."* Possibly this view will change with applications of Concurrent-C Prolog such as in the work by Cleary & al (1988) in discrete event simulation, which may eventually offer the speed that is required to perform simulations.

Also of interest is PROSS (the Prolog Simulation System) as described by O'Keefe & Roach (1987). PROSS is an attempt to provide a discrete simulation system that allows for some modelling of intelligent behaviour. It is based upon an implementation of GPSS within Prolog. The idea is that any AI modelling that can be done in Prolog can be integrated with a simulation model. Ahmad and Hurrion (1988) used Prolog for automatic model building.

Xining & Unger (1986) mentions the advantages of Prolog like languages for simulation :

1. provides declarative semantics based on logic

2. program and data are identical in form : easily manipulated

3. arguments of procedures are not fixed as input or output parameters.

4. backtracking is used to find complete set of solutions.

5. basic elements (atoms and variables) provide a general and flexible data structure superior to the array and records used in other languages.

6. parallel search.

7. programs shorter.

2.5.4 Future Simulation

It is not obvious how all the current research in simulation and its fusion with AI will come together in the end in order to present a new simulation paradigm. However, at this stage already, some conclusions are possible about the difference between simulation of today and the simulation of the future. The major difference will be in the way the model is constructed and run (Shannon 1986, Shannon, Mayer and Adelsberger 1985).

Simulation today is an iterative process in which :

- models are designed;
- inputs are decided upon;
- experiments are run;
- results are analysed;
- other inputs are decided upon;
- another experiment is run;
- etc.

The current simulation systems do not provide aide in deciding upon an appropriate model, nor in how to exercise it to find answers to the problem under consideration. The modeler must translate the logic and behaviour of the system into a set of imperative statements which are then executed from top to bottom in sequential order except when interrupted by control statements.

An AI based 5th generation system would follow a very different paradigm in which the modeller declares the knowledge about the system (especially the description of the objects), defines the goal and lets the computer work to find the solution. It becomes the responsibility of the expert simulation to automatically find the model which fulfills the desired specifications and to execute an appropriate search to obtain the desired solution. In effect Shannon is suggesting a system that would address all the decision problems of simulation.

Another difference between current simulation models and a 5th generation simulation system would be certain characteristics as illustrated in figure 2.3.

	Conventional Simulations are		AI based expert simulation systems would
(a)	Primarily numeric;	(a)	have many symbolic processes;
(b)	Algorithmic (solution steps are explicit;	(b)	use pattern invoked search (solution steps are not explicit);
(c)	Integrated information and control;	(c)	command structure separate from knowledge domain;
(d)	Several steps of modelling process done separately or outside of the model (eg. testing input data for goodness of fit; determining sample sizes; designing the experiment to be run; etc.);	(d)	have all the expertise possible built into the model so that decisions by the user would be minimised;
(e)	Model cannot do anything which is not preplanned (ie the user must instruct the program about what to do).	(e)	the model would be able to learn from its own experience and modify itself as needed. In general, AI systems cannot do this yet although it is theoretically possible.

Figure 2.3 ; difference between current simulation models and a 5th generation simulation system.

Shannon (1986) points to a few more desirable features. The simulation system should:

- allow the user to pose questions and the system decide what kind of experiment to run, required sample size, etc;
- be capable of learning and able to modify itself as it goes along;
- be capable of discovering the optimal schedule or how to reach a specified goal.

2.5.5 Towards the development of support systems in experimental design and analysis

(i) Introduction

From the literature, it is clear that many of the techniques of artificial intelligence will be embedded in the practice of simulation. The goal is to facilitate the use of simulation. *"Today, in order to use simulation correctly and intelligently, the practitioner is required to have expertise*

in a number of different fields... This translates to about 720 hours of formal classroom instruction, plus another 1440 hours of outside study (more than one man year effort) and that is only to gain the basic tools" (Shannon, 1986).

The literature has revealed concentrated efforts to deal with the decision- problems relating to the phases of formulation, programming, verification and validation. There does remain a lack of support for the experimental design and analysis phase of a simulation study.

> *"Expert systems which help with experimentation and analysis are an interesting area for work. Given the increasing trend to hand over simulation models to clients who in the main are inexperienced simulation users, there is a growing need to support the use and guard against the misuse of such models "* (O'Keefe, 1986b)

(ii) Advice-giving systems for discrete event simulation experimentation

O'Keefe (1986b) presented a pilot system, TRANS, that advises on experimentation with transaction models. The aim of the consultation is to help indicate which resources, transactions or activities are interesting candidates for possible experiments and what statistics should be collected given a certain design objective. In parts of the consultation the system assumes that a number of simulation runs have already been tried, and thus comparison can be made between certain output statistics.

Possible objectives were :

- increase utilisation of a resource;
- decrease utilisation of a resource;
- reduce the throughput time of a transaction.

The system was implemented in ES/P Advisor Shell.

O'Keefe classified knowledge used by the experimenter when he determines an appropriate design that satisfies one or more objectives :

(1) knowledge about the domain in which the model is built;

(2) knowledge about statistics : how to interpret results, what measurements are appropriate;

(3) knowledge about how simulation (and hence the real system) behaves;

(4) knowledge of the language or package used to implement the simulation.

TRANS utilises knowledge of type (3) only.

He concludes that simple simulation advisory systems associated with a particular simulation package and/or application domain can be quite easily developed now.

(iii) Experimental Frames in MODELLER

Luker (1986b) presents MODELLER, a software system for the computer assisted modelling of continuous systems. Luker (1986b) has the objective of providing interaction at run time and assisting with the design of experiments... MODELLER is a *"comprehensive software environment to support continuous system simulation"*. *"Modeller was given its own simulation execution system for conducting simple experiments"*.

Five experimental frames are catered for, from a single run to a parameter optimisation, which provides an algorithm for maximising or minimising a given cost function by automatically adjusting designated parameters. The user may specify which frame should apply.

(iv) Experimental Frames in SIMAN

The SIMAN modelling framework (see Pegden, 1984) is based on the theoretic concepts developed by Oren and Zeigler (1979) which separate the system model from the experimental frame. The experimental frames define the conditions under which the model is run to generate specific output data. For a given model, there may be many experimental frames resulting in many sets of output data. By separating the model structure and the experimental frame in two different elements, different experiment can be run by changing only the experimental frame. The model remains unchanged.

In SIMAN, the data analysis is performed after the development and running of the model and is completely distinct from it. One output file can be subjected to many different data

treatments without re-executing the simulation program. Data analysis may also be applied to multipe sets of output files: this is useful to compare the system's response under two or more parameter configurations. In effect, the SIMAN experimental frames support the user who knows which frame to use and what results he wishes to consider.

(v) An Expert Systems for the analysis of output

Haddock (1987) added to the SIMAN environment by building a model-generator. He proposes a framework for an expert system to analyse the output of experiments. Haddock reported O'Keefe's distinction of knowledge used by the experimenter and described a system that also required only knowledge of type (3) from the user. His system, however, did possess facilities to act as if it possessed knowledge of type (2). Indeed, Haddock stressed that in making simulation available to a broad range of users, statistical analysis of the simulation output should be incorporated within the system, thus avoiding the misuse of the model and reducing the time required to perform the analysis. In his system, the output analysis is accomplished by Fortran written subroutines which are incorporated within the software structure of SIMAN, interpret the results of experimental runs and make statistical inferences about the performance measures.

(vi) Automatic Analysis in Knowledge-Based Simulations

Reddy & al (1985) considered automating the analysis of simulations in the context of KBS. They point to conventional simulation languages where data collection processes must be "hard-wired". They then introduce the idea of selective instrumentation whereby only data relevent to a particular goal may be gathered or analysed. It becomes possible to specify an organisational goal and to define a series of constraints which represent subgoals. The ultimate objective would be to allow a user to simply specify a goal and have the system perform. What has been achieved however is some degree of automation in the analysis procedure which permit to go beyond the stage of presenting what happened to why it happened. The amount of modelling that is required in terms of the possible goals, limits the level of support that can be currently be obtained.

2.6 Objectives and Reasons for this Thesis

2.6.1 Need for Experimentation Support

In chapter 1, a concern was raised about the problems of experimental design and analysis in simulation. Throughout this chapter, the objective has been to identify what support is in fact available and failing that, in what context support can be provided. The theory behind simulation studies was investigated and a look was taken at how support was provided in the decision-problems of simulation. The following observations may be made.

Way back in 1959, Conway & al were stressing that *"Perhaps the first lesson the neophyte simulator must learn is that research entirely by experimental methods is a painfully slow and difficult process even under the ideal conditions of control which simulation provide"*. In 1968, Mize & Cox were still preaching that *"Proper considerations of statistical design methods can lead to significant reductions in computer time, and more importantly to improve interpretation of data gathered from simulation experiments"*.

It is noted that, over twenty years later, there is still no proper support for the phase of experimental design and analysis. Grant (1986) *"Simulations have to be interpreted by skilled OR scientists before a naive user can readily understand them"*.

Eventually, non-academics as personalised by Smith (1986) express their alarm at the dangers of misuse of simulation and the lack of support. Academics are not all surprised. Zeigler & De Wael (1986) *"Most simulation practitioners have not had such exposure and consequently the results of simulation studies are frequently suspect from a methodological point of view"*. Solomon (1982) related her experience : *"Some senior people in the field believe that statistical analysis does not prove much, but demands a great deal of effort"*.

The popularisation of simulation prompts Mathewson (1985): *"As more users experience the ease with which models can be built, I believe that we shall need to strengthen training in the proper use of simulation"*. This is justified by Moser (1986) who writes *"Simulation does not provide solutions, it merely shows values of a set of variables over a period of time given certain*

assumptions. Interpreting theses values and drawing inferences from them lies beyond the scope and intent of simulation. Nevertheless, interpretation and inferences are of critical importance to the user. Furthermore, the interpretation and inferences of simulation output is often complex and remains the exclusive domain of experts in many cases". But even experts require assistance as Luker (1986a) notes *"An expert needs more support than a language can provide; there must be an integrated software environment to support all stages of the simulation process from requirements specification to analysis of the results".*

Support for the phase of experimental design and analysis is still in proposal form as suggested by O'Keefe (1986b) *"Given a domain expert with little knowledge of simulation, a useful advisory system for the experimental design phase might give advice on the statistics of conducting experiments".*

2.6.2 Limitations of Current Research

As was discussed in the previous section, some authors have addressed this problem. Most significantly O'Keefe, Luker, Haddock and Reddy & al.

Luker and Haddock have presented support in the context of program generators. They have built a number of experimental frames that support experimentation in its execution. However, the user is still entirely responsible for the input and selection of appropriate frames. There is no support in this respect. The higher level strategies to address specific problems through experimentation is not supported either.

O'Keefe only addressed a small example problem of experimental support. His system was very limited in scope. It did not offer any support in the actual execution of experiments.

The work of Reddy & al offers more insight towards higher level strategies of experimentation. Their work appears strictly limited to KBS models, since instruments, goals and constraints have to be constructed manually for each new application. This application is still geared towards the expert user in its current implementation.

2.6.3 Proposed Research

There does not appear to be any doubt about the importance of support for experimental design and analysis. Many authors propose such support.

Balmer & Paul (1986), in their CASM project, talk about "*statistical design and analysis issues which seem to be specific to systems simulation, including the proper employment of variance reduction techniques, the determination of appropriate run-lengths and the avoidance of bias due to transients*".

Reddy (1987) on knowledge-based simulation suggest that the ideal KBS should among other features be able to accept a goal in the form of expectation and :

- select model;

- determine performance metrics;

- generate search space of plausible scenarios;

- execute simulation model by controlled selection of scenarios;

- recommend a scenario that satisfies the stated goal.

Moser (1986) proposes the creation of an integrated decision-support system developed by combining the ability of expert systems, which allows them to analyse the simulation output and draw the necessary inferences.

According to Oren (1986a) in "*Knowledge-bases for an advanced simulation environment*", an experimentation advisor has several functions:

1. specification of experimentation (including specification of variables to be observed, instrumented and monitored.

2. link a model to different experimental conditions.

3. sensitivity analysis

Whilst talking about directions to explore in artificial intelligence, Oren & Zeigler (1987) suggest knowledge-based intelligent systems which have an ability to perform simulation studies that can define several scenarios within which they can simulate a system to increase their knowledge

about:

- behaviour of the model;

- sensitivity of the behaviour of a model to parameters or operating conditions.

Lehman & al (1986) also considers a system in knowledge base modelling where the results of experimental applications are statistically and graphically analysed in a dialog with the user. This system however is still very much in proposal form.

2.6.4 Objective of this Thesis

The objective is to research into the development of a support environment for experimental design and analysis by considering the use of artificial intelligence techniques such as production rules, heuristics and expert systems.

From Shannon's (1986) analysis of future simulation it appears that there is no certain outcome about the future of simulation modeling. However, there will clearly remain a need for support to experimental design and analysis. In due course, this may well be integrated into the environment where the model is developed. For now, it is proposed to build a support environment that accepts Zeigler's contention (1976) that the model is to be separated from the environment where it is run. Such an environment would permit transportability from one simulation implementation to another.

Experimental design and analysis will remain a difficult task. It is not the objective of this thesis to resolve the existing problems by developing new expertise, but only to consider a framework whereby the existing expertise may be made more accessible to support the simulation user. It is considered that this expertise may go beyond the application of simulation techniques and reflect expert behaviour in the experimentation process.

Further interest could then be satisfied such as considered by Phelps (1986) "*Of interest are ES that will carry out statistical work normally done by statistical specialists*". Also, it could be considered capabilities of a system as described by Martins (1983) "*The simulator should be able*

to make inferences from observed events to some unobserved effects".

Knowledge could be shared by making it available in the form proposed by Davis and Lenat (1982) by allowing why, how, explain questions. Alternatively further knowledge could be gathered from experts about the tactical issues of simulation.

The implementation would be in Prolog as there appeared from the literature much encouragement about its potential in addressing a variety of simulation issues using AI techniques . Additionally, Adelsberger believes that *"the simulation expertise (running and designing experiments) can be performed in Prolog automatically"*. Shells will not be used since they are often found inflexible (d'Agapeyeff, 1984; Blakemore & al,1986) and this would constrain the research horizon.

End of Chapter

Chapter 3

An intelligent controller - Human intelligence & Machine control

3.1 Introduction

The previous chapter stressed the value of research into the development of a support environment for the phase of experimental design and analysis. In this chapter, the interest lies first in an attempt to formalise the action process involved in the execution of this phase of a simulation study. The examination of a typical experimentation environment, through a case study example, uncovers the author's original motivation for this research and provides a general understanding of the action process involved. It is postulated that this process requires a combination of (non-procedural) reasoning and (procedural) control of execution. The reasoning component appears in the form of intelligent specification of the experiment to be carried out. The control aspect deals with the execution of experiments in accordance with the specification obtained from the intelligent component. These components can be characterised by their nature. Typically, they have a human nature ie. they are performed by humans. An appropriate methodology appeared to be to investigate how much support may be provided to a simulation user if some aspect of these two components receive a machine nature. The rest of this chapter considers how the control of execution may be handed over to a machine.

Section 3.2 presents a background case study which exemplifies the conventional approach to the experimental design and analysis phase where the action process is characterised by human intelligence and human control. In section 3.3, a formalisation is proposed of the experimentation process by adopting a procedural framework. In section 3.4, a means of separating the procedural control from the non-procedural input is presented using two machines linked via a serial RS232 interface. Section 3.5, suggest the first steps towards non-procedural control and the possibility of integrating machine intelligence into the process.

3.2 Conventional Experimentation Environment

- Human intelligence & Human Control

The underlying assumption is that the simulation model, which is to be used for experimentation, has been formulated, programmed, verified and validated. These stages in the development cycle of the model will most certainly have been executed by a simulation analyst. This analyst possesses simulation knowledge which may be considered as expertise. In fact, in the context of a simulation study, this analyst may be referred to as the "expert". Possibly this "expert", who built the model, will perform the experimentation. Equally likely, the "expert" will hand the model back to an "end-user" for experimentation. This "end-user" is unlikely to have the same degree of expertise and could therefore benefit from the assistance of the simulation analyst (the "expert"). Unfortunately, the "end-user" - who will from now on simply be referred to as the (simulation) user - may too often be left alone to deal with the design of experiments and analysis of results. In the following, a model is presented for the sake of reference and a case-study example reveals the dangers of naive experimentation behaviour. Expert experimentation behaviour is then considered, leading to a postulate about successful experimentation.

3.2.1 Example Model : the Die-shop

(i) The system

A die-shop model is presented in this section. It is a slightly modified version of an industrial application (Taylor, 1985) and will be used as the reference example throughout this thesis.

The model is developed on a visual interactive simulator. It presents the simulated operations of a computer controlled die-shop where hollow dies are manufactured. There are essentially three types of machines which carry out the processing of hollow dies. These are machining centres (MC), vertical EDMs (VEDM) and wire EDMs (WEDM)). The machining order and processing times of dies are input to the computerised control system, which directs activities and

operations in the shop.

As seen in figure 3.1, linking the machines is a one-way monorail on which a trailer moves clockwise carrying a single pallet loaded with a single die. On the loading platform, dies are manually placed and removed onto pallets. Each die is fixed onto one pallet and will not be removed - even during processing - from this pallet until it returns to the platform to be unloaded by an operator. Also present are carousels which permit the storing of dies being processed but awaiting a machine which is currently in operation. The loading and unloading of pallets on to the trailer is also automated. The automated computerised control centre determines when and where to move the trailer to collect a die, where to transport it, and controls the length of processing by each machine for each die.

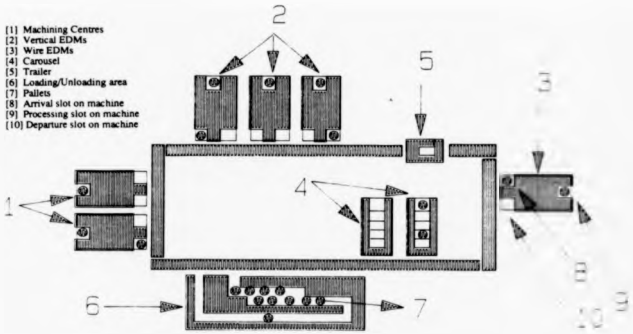


Figure 3.1 : The die-shop model

The first rule which the computerised controller applies is that a die will not be moved from the loading area unless it can enter the machine which will first process it. The second rule is that it will clear dies from machines before bringing new ones to it. Thirdly, the unloading of the carousel is on a FIFO system for each given destination.

It should be noted that a die cannot move between machines unless it is transported by the trailer, and that the trailer can only carry one die at a time.

Each machine has three positions for a die on its pallet. When the die arrives on the trailer to be loaded on to a machine, it enters the "arrival" slot. Only one die can be in the "arrival" slot at any one time. The die will then move to the "processing" slot only if this is free for processing. On completion of the processing, the die will move (again only if the slot is vacant) to the "departure" slot, where it will await the trailer to carry it to its next destination. It is, therefore, possible to have three dies in one machine at a particular moment.

The model allows on screen inspection of the shop and displays movements as dies progress from one machine to another. The path of a die is as follows:

stage 1 : die is loaded onto a pallet by operator in loading area.
stage 2 : die is loaded onto trailer
stage 3 : die is carried on trailer to "arrival" slot of first machine
stage 4 : die moves to "processing" slot of first machine
stage 5 : die moves to "departure" slot of first machine
stage 6 : i) die to another machine and process is repeated
or
stage 7 : ii) die moves to carousel, waits
stage 8 : die is taken on trailer to next machine and process is repeated
or
stage 9 : iii) die is unloaded onto unloading area
stage 10 : die is removed from pallet

(ii) Assumptions:

- each machine is assumed to be continuously available

- no set-up time for operation

- travel time is negligible

(iii) The Problem

The actual plant had one machining centre, one VEDM, one WEDM and two carousels. One of the objectives of the model was to assist in determining desirable alteration to this configuration given different criteria. The problem became 4-dimensional with the following

parameters:

IMO - the number of machining centres

IMTW - the number of VEDMs

IMTH - the number of WEDMs

ICAR - the number of carousels.

The input data is the processing times and routes of the dies. They were assumed to be continuously available and are randomly generated using a single seed.

3.2.2 Naive Experimentation Behaviour - Case study

When the original die-shop model was built, the author was asked to assist his manager in presenting and selling it to the client's board of directors. For this purpose, the author prepared a file of deterministic input data regarding the processing times of a number of dies. This was done to prevent extraordinary happenings during the demonstration. Unfortunately the data was such that after only 500 simulated time units, the second carousel had not been used. This prompted a board room discussion about who was to blame for the expenditure on a carousel which was not needed. In order not to confuse and risk discrediting the model, the author remained silent as did his superior for presumably the same reasons. However, after the presentation when discussing the dangers of misuse with his manager, the author was upset to notice that little regard was given to simulation concepts such as steady-state and replication. The "feel" and understanding provided by the visual component of the model appeared to override the need for any statistical experimentation.

In retrospect the author believes that the visual aspect of the model ensured the acceptance and sale of the model because the client could "see" ie the visual aspect validated the model in the eyes of the client.

However, after such display of naive experimentation behaviour, the question had to be raised about the chance of these non-simulationists to subsequently use their model correctly in

order to obtain results which can be used as a basis for important decisions. There was no evidence of consideration of issues such as steady states or significance. The experimental design and analysis phase would probably be limited to deciding to run the model, running and stopping it and looking at results.

Though the author would not like to generalise this example as typifying the behaviour of "end-users", there appears in the literature references suggesting (the risk of) similar behaviour (see for example remarks by Solomon (1982) in section 2.6.1). The motivation of this research is born out of a concern to provide the "end-user" with assistance comparable to that of an "expert".

3.2.3 Expert experimentation behaviour

Observation reveals that typically an "expert" will :

- consider the model he is to experiment with;
- define task/objective of his study;
- design an experiment to yield results that he expects will help his investigation;
- execute this experiment, in the form of simulation runs, by carefully inputting the experiment configuration values (parameter values, starting and stopping conditions, number of replications, etc) and collecting results;
- perform some preliminary analysis that may lead to possible replication;
- analyse his results;
- present his conclusions.

The expert deals with many points of concern. The design of any experiment considers issues relating to how to achieve a steady state and how to obtain statistically significant results. The analysis of the results of the first experiment may prompt the "expert" to refine the experiment or design further experiments before he can come to a sensible conclusion.

3.2.4 Postulate for Successful Experimentation

It is postulated that successful experimentation requires a combination of (non-procedural) reasoning and (procedural) control of execution. The reasoning component appears in the intelligent design and specification of the experiment to be carried out. The control aspect deals with the execution of experiments in accordance with the specification obtained from the intelligent component. These components can be characterised by their nature. In a conventional environment as exemplified above, both the reasoning and the control have a human nature ie. the experimentation environment has human intelligence and human control of execution.

Dangers of misuse and need for support may appear at both levels. Indeed, if intelligence is equated with simulation knowledge and expertise, the quality of experimental design by non-experts must be suspect. Equally problematic is the risk that mistakes may be made in the control of the experiments designed.

An appropriate methodology appeared to be to investigate how much support may be provided to a simulation user if some aspect of these two components receives a machine nature. This thesis examines different possibilities. The first possibility is to force intelligent input by formalising the procedure to control the execution: ie giving the control to the machine. This is discussed in the rest of this chapter. The second possibility is to give intelligence to the machine and get the human to follow instructions. This is the topic of chapter 4. The last possibility is to remove the human involvement as far as possible from the experimentation process by giving the machine intelligence and control. Implementation of this is found in chapter 5.

3.3 A framework for procedural control of execution

3.3.1 Objectives

In this section, consideration is given to the procedure of execution of simulation experimentation. The approach is to consider how an "expert" might physically carry out an experiment. By generalising his behaviour, an experimental frame can be devised. Requirements for the execution of this frame are outlined. The frame is then automated by devising a procedural controller. The objective of this section is to consider the feasibility of developing a framework for the automation of any number of subsequent frames of expert behaviour and investigating the level of support that such a framework might provide to the simulation user.

3.3.2 A frame for Experimentation

(I) Expert behaviour

An "expert" uses simulation experimentation to obtain results that will help him investigate some objective. The design of such experiments requires intelligence. However, once an experiment has been designed, it may be observed that the process for actually obtaining these desired results involves a recognisable execution process. An example is as follows.

The behaviour of the expert considered (see Kelton, 1985) reveals the design of his experiments always considers:

- what is to be measured
- the model configuration to be used
- the run-in time
- the run time
- the number of runs

This expert executes his experiments according to the following pattern by:

- running the model a given number of times according to input

- collecting results

- deciding if more results are required

- repeating or leaving

- presenting results for analysis

Determining the answers to the above considerations involves intelligent reasoning. Ensuring that these issues are addressed and that the different experimentation steps are executed involves control. In can thus be argued that the execution of an experiment is conditional upon some experimental input. This input is the translation of the specification obtained from the design phase.

Consistently applying this execution process reflects expert behaviour. This execution process can be regarded as a frame for experimentation. Support for a non-expert would be provided by a framework that controlled the execution of this process ie. if this non-expert was forced to follow this pattern every time he wishes to experiment with the model.

The objective of the frame above is only to illustrate that regardless of the model under consideration, an "expert " may replicate his behaviour time after time. It should also be noted that the use of the term *experimental frame* extends the definition provided by Zeigler (1979) who considers it to refer to the limited set of circumstances under which the system is to be experimented with. Here, experimental frame also considers the processes by which these limited set of circumstances are obtained (see Kettenis (1988) for a similar consideration).

(ii) Requirements for simulation experimentation

The following list outlines the basic requirements found necessary in order to execute a simulation experiment, and in particular the frame considered above :

- a verified and validated model

- some set of variables to measure;
- a measuring device and a means of recording
- some particular set of conditions (parameter values)
- a means of determining appropriate starting conditions
- some experimental design procedure
- some means of replicating the experiment by changing the stochastic input
- some means of analysing results
- some output facilities
- some criteria for evaluation

Many simulation packages have facilities that will satisfy these requirements, but in order to ensure generalisation and independence of the simulation implementation used, many of these requirements will be examined from first principles. In some cases, this has involved some development work.

(a) A verified and validated model

The basic assumption in performing a simulation experiment is the existence of a model with which to experiment. This work assumes the existence of a verified and validated model. When examples are required, this thesis will refer to the die-shop model for illustration. The simulation system used is the Fortran-based visual simulation package MICROVISION. The choice of such a system is prompted by the fact that the interest does not lie in developing models , but in using them. Fortran systems (as representative of procedural language systems) are well tested and perform satisfactorily. MICROVISION is the package used at Warwick University's Business School for undergraduate and postgraduate simulation courses.

(b) Some set of variables to measure

Some quantification aspect is necessary: the exact statistics required from a model depend

upon the study being performed, but Gordon (1978) noted several commonly required statistics. These relate to the different components of a simulation model. This thesis considers models to involve :

- entities which are elements that proceed through the system. In the die-shop example these would be the different die components. A useful statistic concerning entities is their average time-in-the-system.

- service queues which represent activity periods. In the die-shop, these would be the queues that represent a machine being in use: these are the processing slots. A common statistic is the average utilisation of these queues as this gives an idea of the usage of service facilities.

- buffer queues which represent waiting queues. Normally, concern would lie in determining the occupancy as a distribution of queue lengths. In the die-shop, the occupancy of the carousels may be considered.

- performance measures is a general umbrella which may take in all other form of statistical collection. These would normally be defined for each model. The work involved in this thesis will consider only statistics of average counts. In the die-shop, the number of parts of dies produced can be considered.

(c) A measuring device and a means of recording

In this thesis, three different approaches are considered to measuring the different quantification variables:

- Package defined: a number of packages have sophisticated devices for computing and measuring the different quantification variables described above. In this work, the average time-in-the-system of entities are available from the MICROVISION package.

- Analyst defined: the model builder prepares these variables during the development stage. The performance measures such as the number of die parts processed needed to be defined by the analyst.

- Research developed: it may be necessary to add to the package in order to provide simulation users with a given type of quantification variables. The utilisation of service queues was added to the MICROVISION package. This measure was defined to be the percentage of time that a given service was in use relative to the total time of the length of the run considered. This was measured as the ratio of the total time in use over total run length. (This was implemented by setting up equivalence classes and using the high level FORTRAN subroutines UTILIS, QUEUES, QUTWES described in appendix 7-d and in the WES technical documentation manual).

The reason for considering these different measuring devices was to obtain indications about the work required for possible extensions of the system that is to be developed in this thesis

Results of runs must be recorded so that a sample of performance measures may be analysed at the appropriate time. Storage of results was initially implemented by using standard output files.

(d) A particular set of configuration values

The term configuration is understood to be defined as a set of values to the parameters of the problem. Interactive models have a series of variable parameters. These parameters may refer for example to resource levels, service times or arrival patterns. Given a particular problem, a subset of these parameters can be of interest. For example, in the die-shop problem, the interest lies with the parameters which determine the number of machines of each type and the number of carousels. These parameters are:

IMO : number of machines type 1 (MC)

IMTW : number of machines type 2 (VEDM)

IMTH : number of machines type 3 (WEDM)

ICAR : number of carousels.

(e) A means of determining appropriate starting conditions

Depending on whether or not the system is terminating, there is a need to consider methods of eliminating or reducing the effect of initial bias. Two general approaches can be taken to remove the bias: the system can be started in a more representative state than the empty state, or the first part of the simulation run can be ignored. (Gordon 1978).

Simulation packages have different facilities to implement the first approach. The second approach appears more general in principle. It is easier to turn recording on after a given period. Rules for determining the appropriate run-in time are package independent. A simple heuristic rule might be to always run a model in for 500 time units.

(f) Some experimental design procedure

Once a stochastic variable has been introduced into a simulation model, almost all the performance variables also become stochastic. The values of most, if not all, of these performance variables will fluctuate as the simulation proceeds, so that no one measurement can be arbitrarily taken to represent the value of a variable. Instead, many observations of the variable's values must be made in order to make a statistical estimate of its true value. Some statement must also be made about the probability of the true value falling within a given interval about the estimated value. Such a statement defines a confidence interval. Without it, simulation results are of little value to a decision maker.

One way of obtaining independent results is to repeat the simulation. A general procedure (see figure 3.2) has been described by many, including Gordon (1978) and Haddock (1987).

Throughout this thesis, simple heuristics are used in preference to implementing complex algorithms. The justification is that algorithms are time-consuming to develop and the resultant values provide little extra advantages over the heuristic values in terms of illustrating framework principles.

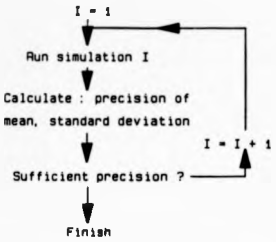


Figure 3.2: A general experimental design procedure

(g) Some means of replicating the experiment by changing the stochastic input

Simulation models have a stochastic input usually. For example, the processing times of dies in the die-shop model follow a normal distribution. Processing times are generated randomly from a given number stream. Changing the number stream would change the stochastic input. The die-shop model uses one random number stream. By making this number stream variable, the objective can be achieved.

(h) Some means of analysing the results

At the most simple level, this might involves computing the lower and upper bounds for the means of the observations obtained from the different runs.

(j) Some output facility

This could consist of presenting mean values with lower and upper bounds of the quantification variables that were selected for monitoring. Use of standard output files are considered for storage of results.

(k) Some criterion for evaluation

At this stage, this lies beyond the scope of the frame and can be considered to be the domain of the intelligent reasoning component.

3.3.3 Automating the experimental frame

In the previous section, the requirements for the execution of simulation experimentation were examined. It was found that, conceptually, they can all be addressed in a formal environment. In this section, attention is returned to the experimental frame identified during section 3.3.2.i

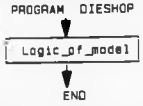
Since the underlying assumption is the existence of verified and validated models, a concern has been to devise a support framework that involves minimum changes to the structure of the model and in no way affects the logic of the model and its validity. The approach has been to implement the necessary changes and then consider how much can sit on top of the model.

(i) Implementation

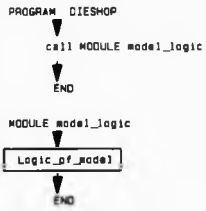
As postulated, the control of the frame involves (intelligent) input and straight forward execution. In this early implementation, there is no attempt to give the system any intelligence, even at the level of preliminary analysis.

a. Running the model several times

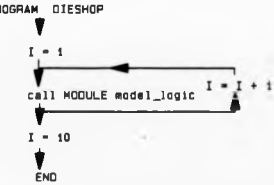
To permit multiple execution, it appeared necessary to alter the structure of the simulation program. Normally a program would contain the whole logic of the problem from the very top. This would appear for example as:



The approach undertaken is to force the following structure:



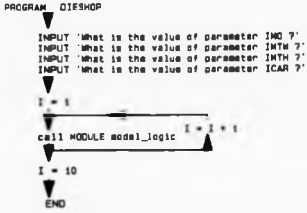
The advantage of this approach is that it becomes theoretically possible to manipulate the execution of the model.



This will permit running of the model ten times.

b. Specifying Input

The user can also be made to decide upon the configuration which is to be used by directing him to input parameter values:

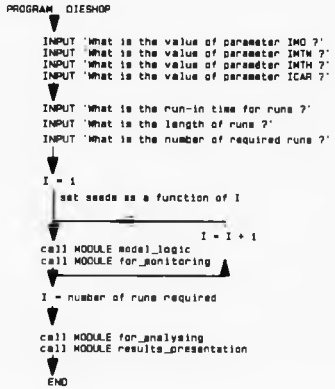


Similarly, the user may be forced to input the run in time, the run length of runs and even the number of runs.

This framework ensures that the model is run according to the specification of the experiment design. No input parameter is ever forgotten. Next the execution process of the frame is automated.

c. Control of execution

To control the experiment, some extra MODULES are required. It is necessary to monitor (i.e. collect results after each run) and analyse the results on completion. Furthermore, it is necessary to change the stochastic input after each run by changing the random number stream. The program structure becomes



The implementation of this conceptual framework ensures that:

1. all design characteristics (parameter values, run in times, run length and number of runs) have been addressed and input into the experimentation process.

2. that the model is effectively run the correct number of times.

3. that the stochastic element is different for each run by making the seed values a function of the run number.

4. that results are collected correctly at the end of each run by a monitoring device.

5. that results are analysed at the end of the experiment.

6. that results are presented.

d. Monitoring, Analysing and Result presentation MODULEs

A quick consideration is given here to the possible implementation structure of the monitoring, analysing and result presentation modules. At this stage, the quantification variables can all be assumed to be user-defined. That is the model builder will have prepared for some summary measures to be monitored. The system can be made to pick up these performance measures using the MONITORING module. Results are stored and collected in array form mono(X,Y) where:

X is the performance variable identification,

Y is the run number

mono(X,Y) is the value of the performance measure X for run Y.

The user would set up this monitoring as suggested below:

```
MODULE for_monitoring
    retrieve the run number Y
    mono(1, Y)  = IBB
    mono(2, Y)  = IBD
    mono(3, Y)  = IDP
END
```

The variables "mono(I,Y)" are then treated generally by the system and can be analysed by the

module ANALYSING which returns an average value mean(X) that can be presented using a module RESULTS. These value would be stored in form mean(X) where:

X is the performance variable identification

mean(X) is the mean value of this performance measure X.

The user could be made to incorporate a module which resembles:

```
SUBROUTINE RESULTS
      WRITE'The average value of IBB is'
      WRITE mean(X)
      WRITE 'The average value of IBD is'
      WRITE mean(Y)
      WRITE 'The average value of IDP is'
      WRITE mean(Z)
RETURN
END
```

A note should be made at this stage about the expected distribution of the quantification variables. The work presented in thesis limits itself to the analysis of mean of means. The reason is that such values tend to normality and can thus be treated using standard Z and t statistics. The quantification values are mean average time-in-the-system of entities, mean average utilisation of service queues, mean values of counts.

(ii) Assessment

The frame presented in 3.2 can be automated as illustrated above. The outcome is that a framework of procedural control has been imposed on the execution of this frame. Practical illustration was achieved by implementing this framework for the MICROVISION package. Changes were necessary to simulation system source code to permit automatic execution of several runs (see appendix 3b).

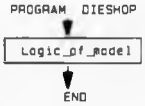
Although the changes in the program are not substantial in the case of a single frame implementation, the user is, however, forced to include code of extra subroutines. This is inelegant and also opens up the risk of error. Furthermore, if other frames were implemented, the model builder would have to program them each time in full for each model considered. To overcome this problem, the next section deals with distinguishing between model-dependent processes and

model-independent processes. The objective is to enclose as much as possible the control mechanisms in a module that would be generally retrievable for use with any number of models.

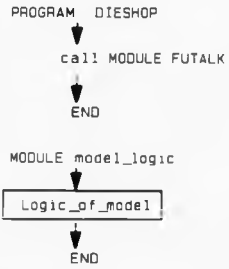
(iii) Model-Dependent & Model-Independent Processes

The implementation used is a Fortran based simulation package called MICROVISION. Models are prepared using high level subroutines. The model is compiled and its object file is then linked with the object files supporting the simulation package to produce an executable model. The objective, here, is to prepare another file which can remain invisible to the model builder and be introduced at linking time. The desired consequence is to force minimal changes to existing models.

Existing models can be changed at their top level from

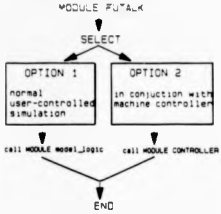


to

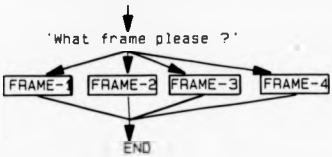


This change allows for a variety of control procedures to be implemented without any consequence for the model-builder. Behind the scenes, a CONTROLLER can now be developed as a super structure.

The module **FUTALK (FORTRAN - USER TALK)** offers the option of using the controller.

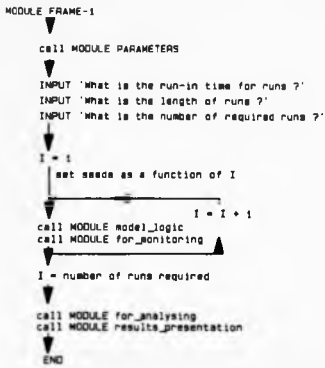


If the controller is to be used, execution passes to **CONTROLLER** which can hold the code developed previously for procedural control. In principle, it is easy to extend the capabilities of the system by implementing other frames. **CONTROLLER** could be used to select a particular frame:



In this case, the system would support three experimental frames, which would be executed by the modules **FRAME_1, FRAME_2 & FRAME_3**.

The frame developed earlier could be expressed:



The instantiation of parameter values is to be taken care of by a module called PARAMETERS. The reason for this approach is that the instantiation of values involves model dependent processes.

```
MODULE PARAMETERS
     INPUT 'What is the value of parameter_1 ?'
     INPUT 'What is the value of parameter_2 ?'
     INPUT 'What is the value of parameter_3 ?'
     INPUT 'What is the value of parameter_4 ?'
     END
```

The drawbacks from separating the model-independent processes from the model can be seen by the loss of model-names. If a general module is to be used for parameter value instantiation ,then parameter values can no longer be requested by name. More significantly in terms of work, great care and effort needs to be taken in linking these dependent and independent subroutines back together during execution. There is a difficult matter of equivalencing variables and parameters in the model with variables and parameters in the independent modules.

The modules manipulate parameters that are indexed param(1), param(2) ... param(n) and general quantification variables ipm(1), ipm(2),...ipm(m). In the model parameters are called by name (for example IMO, IMTW, IMTH) as are the performance measures (IBD, IBB, IDP). These different variable designations need to be equivalenced if the independent and dependent processes are to work together:

IMO with PARAM(1)

IMTW with PARAM(2)

IMTH with PARAM(3)

ICAR with PARAM(4)

IBD with ipm(1)

IDP with ipm(2)

IBB with ipm(3)

In the FORTRAN microvision implementation used, this was resolved by making use of equivalence statements (see appendix 4-b for illustration). The outcome has been that in order to generalise the CONTROLLER implementation, the user now has to make the following changes to his program:

1. Amend the top of his program to become a MODULE.
2. Declaration of equivalence statements to allow parameter values to be instantiated and performance measures to be used by general monitoring devices and analysing routines.

What has been achieved is that through minimal changes a user can now enable his model to be controlled procedurally during experimentation. There is some virtue in forcing the user to declare the appropriate equivalences. It forces him to understand the important parameters and performance measures of his model.

3.3.4 Uses of the Procedural Framework

The framework proposed was used to automate two other experimental frames. The first considers long runs with multiple sets of observations whilst the other implements a general procedure for determining the appropriate run in times. They are briefly overviewed here as they will be used and refined in the following chapters. The notion of a controller is also paralleled with other work. Section 7.2 offers some indications of the experience of using the CONTROLLER.

(i) Experimental frame for long runs with multiple sets of observations

This frame considers the behaviour of an expert who obtains his results by consistently adopting the following process.

1- Expert determines what is to be monitored

2- Expert determines configuration required

2- Expert determines the run in time

3- Expert determines the duration between each set of observations

4- Expert determines the number of replications (sets of observations) required.

5- Expert executes experiment in the form of one long run according to specification required.

6- Expert analyses results

7- Expert presents results

(ii) Monitoring tasks for run in time determination

The general approach has been to observe the utilisation of queues in the system as an aggregate measure. By analysing the trend of the system's average utilisation, a number of conditions are tested to see if it can be assumed that the system has now reached a steady state. The user has to decide which conditions should be tested. Although, the approach is general, the

implementation is specific to the MICROVISION system since modifications have been necessary to the system programs.

(iii) The CONTROLLER notion in relation to other work

Other authors have implemented, within the structure of their systems, experimental frames in a manner that is not too dissimilar (see section 2.5.5 Luker, 1986; Haddock, 1987). The purpose of this framework is explained in the rest of the chapter.

3.4 Enhancing procedural control through remote machine intelligence

3.4.1 Limitations of Procedural Control Framework

In the previous section a framework was developed for the procedural control of simulation experimentation. The idea was to force the user to go through a series of well-defined steps in order to carry out experimentation. These steps were identified when a frame for experimentation was devised. The execution of the frame has been automated and the outcome is a control system which prompts the user for input before executing the chosen experiment. This clearly achieves the goal of developing an environment where experimentation is carried out by machine control and human intelligence. The weakness of this approach lies with the quality of the human input.

An experienced simulation-user may be happy with the prompts from the controller. A naive user may on the other hand feel awkward and would appreciate a little "artificial" intelligence in times of indecision. This suggests two problems. Firstly, how to encode some "artificial" intelligence and secondly how to ensure that the controller knows when to get its input - be it human or machine.

3.4.2 Remote Intelligence

To remedy the second problem, remote intelligence has been considered with a single channel to the controller, ensuring that the input can only come from one direction. It was seen appropriate to consider an AI language for dealing with the intelligent input aspect.

At the time when the above suggestion was formulated, Dr Andrew Flitman was completing work on linking procedural language simulations with a Prolog program (see appendix 1 for elementary introduction to Prolog). Flitman had developed a link connecting two systems which allowed remote communications to a PROLOG program whilst allowing the full interactive facilities of the simulation. The hand-shaking protocol used in his interprocessor link may be found in appendix 2 (see also Flitman (1986)).

The main problem in producing this link proved to be matching data types between pro-

cedural languages and PROLOG. The idea is that the PROLOG waits to receive a command from the simulation. The command consists of a predicate (name) followed by input characters. The command is then executed by PROLOG, while the procedural language (FORTRAN) program awaits its completion. During "command" execution, various forms of interaction between FORTRAN and PROLOG may take place. When all interaction for the "command" is complete, the PROLOG will inform the FORTRAN.

The operation of the link was divided between problem-dependent and problem-independent routines. To facilitate the use of the link, Flitman devised a suite of easy to write problem-dependent routines. These routines were simple to write because they use data manipulation routines included in the link manipulation files.

The link was adopted as it offered several research advantages :

- this work provided validation of the link;
- this work suggested an appropriate application of the link;
- the link appeared to offer an ideal environment to illustrate the separation of intelligent reasoning that is translated into input and control of execution.

The envisaged set-up is proposed in figure 3.3

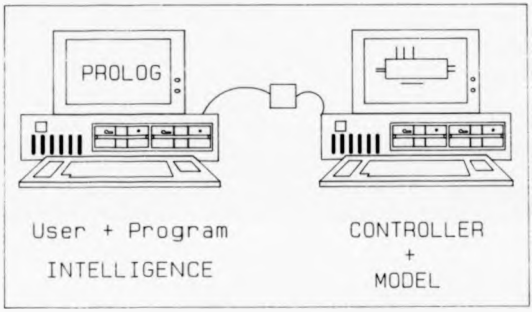


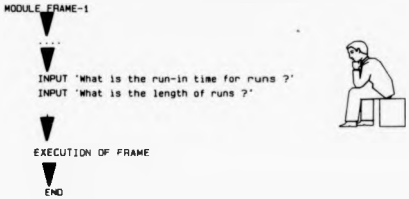
Figure 3.3 : an environment to link the controller to remote intelligence

The input would have to go through a PROLOG program whether input from the user or not. The approach was to develop a number of problem-dependent routines to perform the appropriate tasks. These problem-dependent routines are for inclusion in the procedural language (FORTRAN) side. However, as these routines are only dependent on the experimental controller requirements, these routines, once developed, may sit with the other model independent processes hidden from the user.

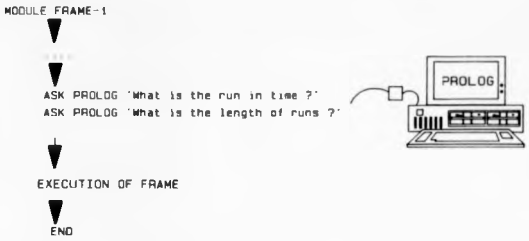
3.4.3 An example

(i) The Fortran component

At this stage, it is useful to consider how this set up may be used practically. In section 3.3.3, an experimental frame was automated in a subroutine FRAME-1. Part of this frame was:



These two questions need to be answered from the PROLOG program. To achieve this, a call must be inserted to the PROLOG program advising it that input is requested regarding run in time and length of runs. The implementation of this call is found in appendix 4-a. Conceptually, the frame now executes as follows:



Once results have been derived, the FORTRAN controller receives them. But how were these values derived ?

(ii) Prolog Program

At the other end of the link, there must be a PROLOG file which contains, at least, clauses for each of the executable commands stored on the FORTRAN side . In the example considered, there must be a clause "frame_1". This clause must derive and send back to the FORTRAN controller values for run in times and length of runs. These values in the first instance could be obtained from the user in the following way:

```
frame_1 : - write('What run in time should be used?')
            read (X), nl
            write('What length of runs are needed?')
            read (Y), nl
            commend,
            writeri(X),
            writeri(Y),!.
```

The clause "commend" indicates that all instructions from the FORTRAN have been received. It prepares the link to send back output from the PROLOG. "writeri(X)" sends an integer value back to FORTRAN.

This simple example demonstrates the possibility of using remote human intelligence to input into a procedural controller. What is still required is evidence that some form of artificial intelligence may assist a user in need of help.

3.4.4 Artificially induced intelligent input

In the clause "frame_1" above, the mechanism for deriving the required values could be removed to separate clauses:

```
frame_1:-find_run_in_time,
         run_in_time_is_(X),
         find_run_length,
         run_length_is_(Y),
         commend,
         writeri(X),
         writeri(Y),!.
```

The clauses "find_run_in_time" and "find_run_length" will carry out an investigation for the required values. These values will be stored in the facts "run_in_time_is_(X)" and "run_length_is_(Y).

Input from the user could be obtained :

```
find_run_in_time : - write ('What run in time should be used ?'),
                     read(X),
                     not (X == 'unknown'),
                     assert(run_in_time_is_(X)),!.
```

There is now a check on the input from user. If the answer to the question is *unknown* , the clause will fail. In the event, the inference engine of the PROLOG implementation will seek for another clause "find_run_in_time" and attempt to satisfy it. If no clause is found or can be satisfied, the original clause 'frame_1' will fail and further back tracking will be required, which in this implementation can only lead to the system crashing.

A first alternative may be to insert a default value:

```
find_run_in_time :-      x is 500
                         assert(run_in_tie_is_(X)),!.
```

This will assert a default run in time of 500 time units. This is of course very basic intelligence. A better approach would be to question the user to see if the system is terminating or not.

```
find_run_in_time :-     write('Is the system terminating?'),
                        read(Answer),
                        Answer == 'yes',
                        X is 0,
                        assert(run_in_time_is_(X)),!.

find_run_in_time :-     x is 500,
                        assert(run_in-time_is_(X)),!.
```

For this example, it is possible to identify three possible values for the run in time of an experiment.

1. X = value from the user

2. 0 from the PROLOG if the system is terminating.

3. 500 default from the PROLOG in all other cases.

Although this is only very low level intelligence from the illustrated PROLOG program, it may be seen that in principle input from the user may be enhanced artificially through the use of an AI language based program.

3.4.5 Assessment

The automated frame described in 3.3.3 was implemented using the link as described in this section 3.4. All input requests are executed through the link. In each case, there is at least the provision of some default value should the user not know what would constitute intelligent input. The outcome has thus been to turn the CONTROLLER into an Intelligent CONTROLLER (see the WES technical documentation for a listing).

At this stage no further changes are needed to the model. However, some modifications have been necessary at the simulation system level. To perform automatic execution of multiple runs, it was necessary to change the execution routines of the MICROVISION system. A parameter IPROL was included to signify the mode of use requested from the package:

IPROL = 0 means execute

IPROL = 2 means execute multiple runs

IPROL = 4 means execute multiple batch runs.

Implementation of these changes to the source program can be found in appendix 3-b.

It was also necessary to modify the link routines to permit execution of industrial size applications (see appendix 3-a). This was predicted by Flitman (1986) as future work.

The dieshop was successfully used to demonstrate the applicability of the process described. In terms of support, the framework described offers the user the chance to submit with minimal changes a model for experimentation according to a number of experimental frames. Execution of these frames is performed automatically.

3.5 Towards a framework for non-procedural control of execution

3.5.1 Limitation of the Procedural Controller

(i) Problem Areas

In section 3.3, a framework was proposed for the procedural control of execution of simulation experimentation and a particular frame was automated requiring intelligent human input. In section 3.4, the procedural controller was linked to a PROLOG program for remote intelligent input. As suggested at the end of section 3.3, the refinement of the controller would probably include the implementation of extra experimental frames. But the process would probably remain the same where :

- user tells controller what frame to use
- controller asks for appropriate input and executes frame.

The control is procedural because after every input there is a logically defined sequence of events.

In this section, consideration is given to cases when the user's input calls for extraordinary response by the controller. For example, if the user did not know what would be an appropriate run-in time, activation of the frame for finding run-in times would be sensible. What is sought is to make the controller more responsive in circumstances when the procedural framework might constrain the intelligent input.

(ii) An example

Suppose that the user has chosen to use the frame developed in 3.4. The focus of attention still lies on the first question of finding an appropriate run in time. In section 3.4 a number of very low level intelligent heuristics were proposed, to provide default values. More intelligent approaches might consider, as suggested above, the execution of a frame for finding run-in times. Alternatively, results from previous experimentation might provide the answer, if these results were stored and accessible.

There are now four possibilities for the controller to deal with a response from the user:

- use input,
- activate another frame,
- retrieve data,
- use default.

The first and last possibilities fits the framework described earlier with procedural control of an automated frame. The second and third possibilities suspend activation of the original frame. The third may even cause abandoning it if results already exist.

It is suggested that the controller needs to become flexible to deal with frames where execution may no longer be procedural. An investigation is carried out into the development of a framework for non-procedural control.

3.5.2 Moving the centre of control from FORTRAN to PROLOG

The first step in developing non-procedural control of execution was to shift the centre of control from FORTRAN to PROLOG. This means that instead of having control of execution with remote input, the situation becomes control and input with remote execution. The implication of this move is that no frame is executed until all required input has been found. This means that although the execution of the frame will be procedural, its control leading to execution need not be. This can be illustrated by the following implementation. (See appendix 4-c for a more technical explanation).

(i) The Fortran side

The FORTRAN controller is replaced by an executioner which waits to be told which frame to execute (see figure 3.4)

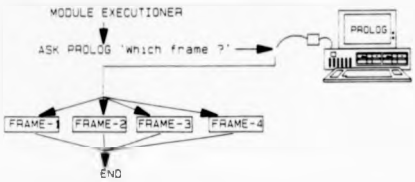


Figure 3.4 : Flow diagram of the EXECUTIONER module

(ii) The Prolog side

The PROLOG based controller selects the frame to the executed. However, it does not send the selection to the EXECUTIONER until it has established that all the required input to these experimental frames can be derived. It controls the input to these frames as illustrated below:

```
controller :- command,
              INPUT "Please select:
                    1. - Run an experiment using multiple runs
                    2. - Run an experiment using long runs
                    3. - Find a run in time
                    4. - Stop "

              read (Answer),

              asserta(current_problem(Answer)),

         asserta(problem(Answer)),

              select_frame,!.
```

The clause identifies from the user the problem to be investigated (problem(Answer)). The fact current_problem(Answer) is stored to identify which investigation is being pursued at the present moment. At any one time, there will be only one "current_problem(X)" under investigation, but there may be several "problem(Y)". These facts are stored sequentially using the PROLOG 'asserta' clause. This allows retrieval in the order input.

```
select_frame :-   current_problem(X),
                  X == 1,
                  execute_frame_1,!.

select_frame :-   current_problem(X),
                  X == 2,
                   |
                  execute_frame_2,!.

      |
      |

select_frame :-   current_problem(X),
                  X == 4,
                  retractall(problem(_)),
              retractall(current_problem(_)),
                  writeri(13),!.
```

According to the problem selected, the clause "select_frame" sends control to the appropriate frame. For example:

```
execute_frame_1 :-
        find_run_in_time,
        find_run_length,
        writeri(1),!.

or
    execute_frame_2 :-
        find_conditions,
        writeri(2),!.
```

The clauses 'execute-frame_X' will find all the necessary input values if execution of a FORTRAN simulation experiment is required. When all necessary input have been found, the FORTRAN is notified about which frame to execute(writeri(X)). Consider the process of finding a run-in time.

```
find_run_in_time :-     run_in_time_is_(X),!.

find_run_in_time :-     write('What run in time should be used?')
                        read(X),
                        not(X == unknown)
                        assert(run_in_time_is_(X)),!.
```

```
find_run_in_time :- write('Is the system terminating?'),
                    read(answer),
                    answer == yes,
                    X is 0
                    assert(run_in_time_is_(X)),!.


find_run_in_time :-     frame_2,
                    asserta(problem(2)),
                    retract(current_problem(X)),
                    assert(current_problem(2)),!.
```

The clause "find_run_in_time" will help determine the appropriate run in period in several different ways:

-1. identified as already known

-2. asked the user

-3. default value 0 if system is terminating.

-4. to be found using another frame (frame_2).

Notice that when frame_2 was called upon (in the fourth clause *"find_run_in_time"*), the execution of that frame became the current problem. The previous "current problem" became suspended. All that is still needed are clauses to feed the relevant input when the FORTRAN subroutines required it. These have form:

```
frame_1 :-  run_in_time_is_(X)
            run_length_is_(Y)
            parameter_1_(A),
            parameter_2_(B),
            parameter_3_(C),
            commend,
            writeri(X),
            writeri(Y),
            writeri(A),
            writeri(B),
            writeri(C),!.
```

As was mentioned at the start of this example, a frame will not be executed by the FORTRAN component until it has been notified that all input has been determined by the PROLOG controller.

What has been described should suggest the ability of a PROLOG program to implement non procedural control for the execution of simulation experimentation. The non-procedural characterisation is justified by its ability to interrupt, suspend (and even abandon - see below) procedurally defined experimental frames in the light of new information (user input or other). Intuitively it makes sense to use the procedural FORTRAN component for procedural execution and the non-procedural PROLOG for non-procedural search and control.

In the illustration given, a brief mention was made about the possibility of retrieving and using results from previous experimentation. In the next section, consideration is given to improving the potential use of results from previous experimentation.

3.5.3 Enhancing input through previous experimental results

Using previously derived results requires the ability to

-store,

-retrieve and search through

-update.

Since the results are derived and analysed on the FORTRAN side, it might appear logical to also store them there. However, as the expectation is that the PROLOG program should do the searching, and in view of the fact that PROLOG as a language has natural database characteristics, work was encouraged in this direction. Consequently, a transfer mechanism was implemented to send results back to the PROLOG side.

a. Quantification variables

Results are clearly model-dependent. Each collected value has a context which must not be lost if it is to be used again. For example a value of 40% might refer to the average utilisation of queue IAMAQ(11) in the die-shop model after a given experiment numbered 1. All these associated facts should be stored with this value of 40%. This could be stored as

fact(queue_1,1,40)

where the user would have to know that:

queue_1 = iamaq(11),

1 means experiment 1 and

40 refers to average utilisation.

A more pleasing implementation is if the name of the queue could actually be used. Parameters and their values must be kept to record the experimental configuration. Frame input specifications should be kept.

By using different modifications of the feedback routine described above, it was made possible to read back after every experiment the analysed results of :

- every queue utilisation,
- every performance measure.

b. Design Specification

Also stored were the input values into every frame. For example, after the execution of a frame, a fact *action* stores this information depending on the input required. For example :

action(Frame,Experiment,Runin,Runtime,Frun,Lrun,Qu,Pv)

was stored, where :

Frame : designated the frame selected,

Experiment : the number of this experiment,

Runin : the run in time required,

Runtime : the duration of runs,

Frun, Lrun : the number of first run and last run,

Qu, Pv : binary indicating whether to monitor queue and performance measures.

c. configuration design

The actual model configuration for this experiment is held in a fact :

configuration(Experiment,A,B,C,D,E,F).

where Experiment designates the experiment number,

A,B,C,D,E,F are the parameter values.

By combining, these facts it becomes possible for the system to checks for the configuration used, and then to look at past experiments to see if, for example, the run-in time has been established previously.

If all these facts can be kept, ready for retrieval and use, then the potential for intelligent input is increased. The foundations are laid for intelligent computer understanding of the model by the controller.

Further intelligent understanding of the model is implemented using the concept of a self-explaining model.

3.5.4 Self-explaining models

The idea of a self explaining model is to make the PROLOG controller aware of the characteristics of the model which is being experimented upon. At the initialisation phase, the model informs the PROLOG about:

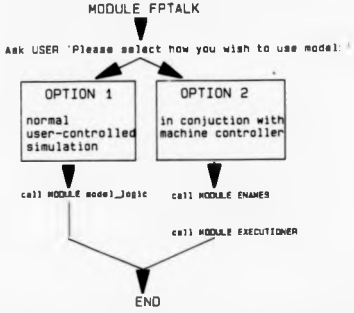
- the entities,
- the queues in the system,
- the parameters,
- the performance measures.

The PROLOG program can set up a database storing the names of the components of the model. When results are to be stored the appropriate names can be used. This means that search by names is possible. The user could request the value of performance measure ibb and the PROLOG program would pattern match.

Concepts of self-explaining models were implemented as follows. (For details, see appendix 4-d). The user was made to include subroutines in his model which read over the names of queues, parameters, entities and performance measures. The PROLOG reads these names and stores appropriate facts to record this acquired knowledge. On the PROLOG sides, the following clauses were devised to pick up appropriate names:

```
enames :-   reada(Atom),
            commend,
            enamenu(X),
            Number is X
            asserta(ename(Number,[Atom]))
            echeck,!.
```

The clauses read the names from the model and associate a "number" fact with the given name. The name can be used when storing results. This name-learning process is activated when the model is initialised through the module FPTALK which now replaces the old FUTALK.



The reading back of results can now be implemented through a couple of subroutines RESBQU and RESBPV and a couple of clauses resbqu and resbpv which are to be executed at the end of frame_1.

3.5.5 Assessment

By implementing the procedures described in this chapter, the dieshop model was successfully turned into a self-explaining model and operated satisfactorily under a non-procedural controller for simulation experimentation. Clauses were added to help retrieve input to frames by consulting results derived from previous experimentation.

3.6 Conclusion

It was postulated that the process of simulation experimentation can be characterised by the control aspect of the execution of the experiment and the intelligent input which determines the experiment. In section 3.2, an example was given to illustrate how in normal circumstances, a human user is responsible for both parts of this process.

In section 3.3, a frame for experimentation was adopted and it was seen how it was possible to enforce a framework of procedural control to the process of experimentation by developing a machine controller of the process with human intelligent input. The weakness of this approach lies in the quality of intelligent input. In section 3.4, methods for enhancing this framework were considered through the use of remote intelligence. Two computers linked via serial RS232 interface permit the separation of the design input aspect from the control aspect. Limitations were then found with procedural control in areas of application of simultaneous execution, suspension or abortion of multiple frames. To remedy this, in section 3.5 a framework for non-procedural control was devised. This allows more flexibility for the controller in the manipulation of frames. An aspect which appeared in this context was the use of the results derived from previous experimentation. Early work was carried out towards enhancing the quality of the data stored. The concept of a self-explaining model was devised to provide the controller with the basic understanding of the model under consideration.

The benefits have been that :

- Through minimal changes in the common file and at the top level of an existing model, with the inclusion of a couple of subroutines, it has been possible to prepare models for use with the controller.

- The controller in its final form offers the automatic execution of a number of frames that mimic expert control of execution : thus insuring that experiments once designed are executed correctly. Execution no longer involves any effort on the part of the user in terms of results collection and basic analysis (determining mean values and confidence intervals)

- By linking the controller to a Prolog program, it has been possible to assist the user with

low level intelligence in the form of simple rules and default values for the input to the experimental frames.

- By later removing the control to a non-procedural domain, it has been possible to make the controller more responsive to input by letting it change its behaviour accordingly. This has provided the controller with the ability to trigger off frames to obtain results to input into frames selected by the user. This increases the support level.

- By making use of Prolog's database suitability, it has been possible to get the controller to make use of previous experimental results. The concept of a perfect memory can be considered.

- By teaching the controller the names of the components in the model, it has been possible to regain some impression of a system that "talks" in domain-language rather than in generality.

This chapter has demonstrated the feasibility of machine control to the execution of simulation experimentation with (low level) intelligent input. The usefulness of the results derived are subject to the appropriateness of the experiment designed. This concerns the intelligent reasoning process that must be carried out in every simulation experimentation. In the next chapter, the concern will lie with an attempt to support the user at the level of the intelligent input component.

End of Chapter

Chapter 4

An Advisor - Machine Intelligence and Human Control

4.1 Introduction

The process of simulation experimentation requires intelligent input and formal control of execution. In chapter 3, the interest lay in developing machine control of execution, while the input was essentially of human origin. Initial work was undertaken to consider implementation of input that may be obtained by other means than direct questioning of the user. Mechanisms were derived to support input from a Prolog program.

In this chapter, a different view is taken. The quality of simulation experimentation may be enhanced if the user is advised on how to control his experiments. Essentially, the aim is to develop machine intelligence to input to a human control process. The machine is to determine what is to be done and the human is to do it.

To achieve this objective, the use of artificial intelligence techniques are considered. The desired goal is to mimic a human expert in simulation experimentation. This involves designing a tool that will behave, reason and inference the way a human expert does. In the field of artificial intelligence, conceptual tools called expert systems (ES) have been devised. In this chapter, the appropriateness of adopting an ES framework for simulation experimental design and analysis purposes is considered (section 4.2), together with a methodology for developing such a system (section 4.3). Thereafter, an account is given of the implementation of an advisory system (section 4.4) , with a detailed look at the areas which are addressed and the support that may be provided (section 4.5).

4.2 Appropriateness of Expert System Framework to Simulation Experimentation

In the second chapter, a review was made of expert systems. The literature revealed that, in the appropriate domains, expert systems can be seen as powerful tools that offer great potential. Some authors have considered the similarities between simulation and expert systems, but this interest has mainly focussed on the model development issues. The objective of this section is to investigate the appropriateness of applying the expert system's approach to the domain of experimental design and analysis. This involves determining the tasks that are to be addressed, and examining the nature of the problems that require assistance within these tasks. The idea is to consider whether these tasks and problems exhibit characteristics found in other expert system domains.

4.2.1 Simulation Tasks

Shannon, Mayer and Adelsberger (1985) argue that the design of a simulation experiment is determined by the goal or purpose of the study. These authors have noted several reasons for simulation experimentation :

- Evaluation of the performance of system against some specific criteria;

- Comparison of different system configurations on different operating policies and procedures;

- Prediction of the performance of a system under a set of conditions;

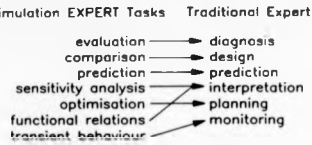
- Sensitivity Analysis to determine which of many factors are most significant in affecting overall system performance;

- Optimisation by determining which combination of factor levels produce best performance of a system;

- Functional Relations to establish the nature of relationships between one or more significant factors and the system's response;

- Transient Behaviour by looking for special bottlenecks, excessive queue buildup or utilisation imbalances.

These seven reasons for making use of simulation tools can be regarded as the expert tasks of the experimental design and analysis phase of simulation. They can be mapped onto the more conventionally understood expert system tasks listed below on the right :



The list on the right refers to the characterisation of tasks that Expert Systems tend to be good at (Hayes-Roth,Waterman,Lenat 1983). The realisation that the needs for calling upon simulation can be broken down into (a combination of) tasks that are not too dissimilar to conventional expert systems tasks prompted further investigation into the appropriateness of expert systems in simulation experimentation.

4.2.2 Nature Characterisation of Simulation Experimentation Problems

Typically, in most expert systems, one or more of the following problems are dealt with :

- unreliable data,
- unreliable knowledge,
- inexact rules for combination evidence,
- fuzzy or inexact relationships.

In attempting to experiment with a simulation model, analysts certainly come across times when they address each of these problems. Indeed, they wonder about the configuration characteristics of their model because the relationship between the number of runs, for instance, and queue buildup appears fuzzy or inexact. The problem of a queue that reaches an excessive length may be remedied by either better scheduling or a different configuration; however the rules for combining evidence are inexact, and to avoid a queue buildup might result in lower throughput. Predictions may be carried out to be checked against empirical results because it is feared that the

data may be unreliable. The need for evaluation might result from concern about the reliability of simulation practice itself (i.e. unreliable knowledge about how to make use of simulation).

Expert systems are used when there is no clear theoretical underpinning, but where the experts recognise a collection of empirical associations without necessarily knowing the reasons for many or all of the relationships. Simulation is such an area. There is no doubt that substantial statistical work has been carried out in the field of discrete event simulation. There are formal ways of designing and running simulation experiments. This was illustrated in chapter 3. There are also well grounded theoretical procedures for analysing the results of these experiments. It is possible to insert these techniques as knowledge into an expert system. The problem comes, however, in deciding which experiment to use in the first place. This is where there is no theoretical underpinning on how this should be done. Analysts recognise that a queue buildup can perhaps be remedied by a better schedule or a different configuration. How an analyst decides on which strategy to adopt to investigate these alternatives is a matter of individual expertise.

Support may be provided to an end-user if this expertise could be encoded for use by an advisory system. This expertise would include the formal techniques of experimental design, statistical analysis mechanisms and strategies for using these techniques and mechanisms in given problem situations. The advisory system would carry out the intelligent reasoning process and the user would control the execution of experiments as advised.

4.2.3 Appraisal

From this conceptual investigation, it was concluded that a theoretical suitability of applying the expert system's framework to the domain of simulation experimentation should exist. Simulation experimentation was seen as performed in a number of tasks that could be paralleled to expert system tasks. The general characteristics of expert system domain problems were noted to exist in simulation experimentation. There also appeared an apparent suitability of the expert systems framework to capturing simulation experimentation expertise. This is further investigated in the next section on the development of a prototype expert advisor.

4.3 Development Framework for an Advisor in Simulation Experimental Design and Analysis

This section considers the issues involved in the development of an advisory system that adopts the expert system paradigm. The essence of an expert system lies in the knowledge it embodies. Its power is directly related to the amount of knowledge that can be encoded in a form that is amenable for use in the decision-making process that the system is intended to support. Therefore a crucial part in developing an expert system is the stage termed knowledge-acquisition. A general approach to knowledge-acquisition is shown below in figure 4.1 (Hayes-Roth, Waterman, Lenat 1983) :

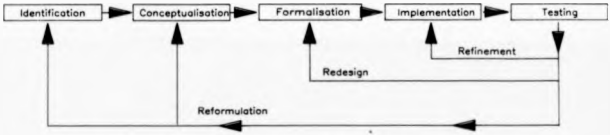


Figure 4.1 : A general approach to knowledge-acquisition

This approach favours the prototype development, and is used in the course of this work. This section examines the conceptual guidelines that this framework provided in the context of simulation experimentation. The nature of this approach implies actual implementation and step-wise growth. This is discussed in the rest of the chapter.

4.3.1 Identification

(i) Objectives

The identification step involves determining the problem domain and its decision-making process, characteristics, and suitability in principle to expert systems. It is also necessary to identify the knowledge engineer and the domain expert(s). These last two points may be resolved immediately. The knowledge engineer will be the author who will also be the initial expert. The

justification for this choice lies in the fact that the objective is only to develop the principle and framework for a support tool. If an acceptable framework is found then it should be possible to incorporate further knowledge ("expertise"). This will be addressed in chapter 5.

(ii) Domain analysis : simple intelligent reasoning model

The initial part of the identification process was addressed in the previous section when the appropriateness of an expert system framework to the domain of experimental design and analysis was considered. Within this domain, the context in which the decision-making process occurs is provided by the behaviour of an "expert" simulation analyst. In chapter 3, the following behaviour pattern was presented where the expert :

(1)- considers the model he is to experiment with;

(2)- defines the task/objective of his study;

(3)- designs an experiment to yield results that he expects will help his investigation;

(4)- executes his experiment, in the form of simulation runs, by carefully inputting the experiment configuration values (parameter values, starting and stopping conditions, number of replications, etc) and collecting results;

(5)- performs some preliminary analysis that may lead to possible replication;

(6)- analyses results;

(7)- presents conclusions.

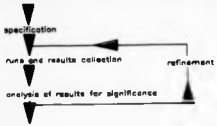
This understanding of the experimentation process was acceptable when the consideration lay with the control of execution of experiments. However, since the focus is now on the intelligent component, it is necessary to refine this model of the reasoning process. The concentration area will be steps (3),(4) and (5)-(6). Essentially there is intelligence involved before running a model that deals with designing the experiment, and there is intelligence involved after the run in the analysis stage. Therefore, conceptually steps (3),(4) and (5)-(6) can be characterised respectively by :

- intelligence in design,

- control of execution,

- intelligence in analysis.

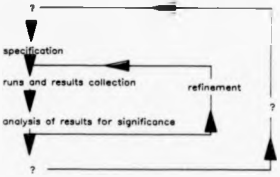
The decision making process can be graphically modelled :



The outcome of this process is that results are always significant.

(III) Domain analysis : extended intelligent reasoning model

The previous model only addresses the specification of one experiment at a time. In general, this pattern is part of a larger investigation that may require several experiments. Extending the previous model implies consideration of what happens before an experiment is specified and what happens after results have been collected.



(a) Intelligence in Design

This phase starts once the task has been defined by the user. His understanding of the model and task will lead him to consider a plan for approaching this problem. This planning will be born out of his simulation knowledge and experience and will be referred to as strategic reasoning. Strategic reasoning may provide at least two types of plans :

- to design a starting plan by specifying an experiment, in order to react to results;

- to design a global plan that specifies several experiments and then to analyse all results together.

The plans will be called strategies.

(b) Intelligence in Analysis

Different levels of analysis are required after an experiment is executed.

- Checking for Significance : which may lead back to refinement.
- Evaluation of Performance : once results have been collected, there is a need to analyse them and report their contribution. This may lead back to some more strategic reasoning.
- Analysis of the Investigation : after all experimentation has been conducted, a more detailed analysis may be required.

(c) Model of Intelligent reasoning

By combining the above, the following model may be derived as presented in figure 4.2.

This translates into the general framework of expert behaviour where the analyst :

(1)- considers the model he is to experiment with;

(2)- defines task/objective of his study;

(3)- considers a strategy;

(4)- designs an experiment to yield results that he expects will help his investigation;

(5)- executes his experiment, in the form of simulation runs, by carefully inputting the experiment configuration values (parameter values, starting and stopping conditions, number of replications, etc) and collecting results;

(6)- performs some preliminary analysis that may lead to possible refinement of this experiment;

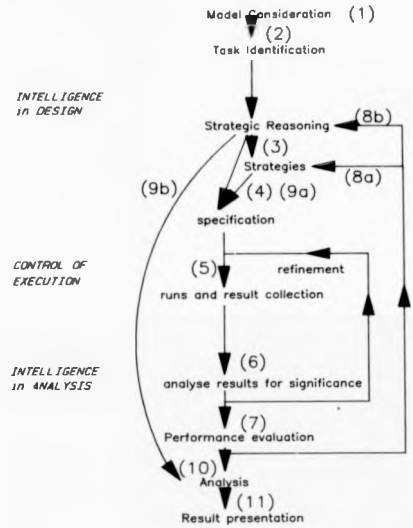


Figure 4.2 : Extended model of Intelligent reasoning

(7)- evaluates this experiment;

(8a)- returns and executes another experiment that was already conceived as part of a global strategy;

(8b)- reacts to evaluation of previous experiment and then consider further experimentation;

(9a)- process returns to step 4;

(9b)- exit from this cycle and proceeds;

(10)- analyses results;

(11)- presents conclusions.

(iv) Decision-Making

In the context of the support framework envisaged in this chapter, where the machine provides the intelligence and the human executes the experiment, the support for the user in the decision-making process occurs at two levels : in the specification of experiments and in the analysis of the investigation. In other words, the output from the proposed advisory system must be a set of instructions on how to set up an experiment and advice about the conclusions that can be drawn after the experiment. The user does not necessarily have to be informed about the higher level strategies which determine the design of the individual experiments.

4.3.2 Conceptualisation

This stage involves identifying the terms and concepts used by the experts in the domain under consideration. These terms and concepts will be used to express the knowledge that is required from the system.

Figure 4.3 presents a list of desirable concepts that are used by experts in simulation experimental design and analysis (Gordon, 1978). Ideally, it must be made possible for an (computer) expert (system) to understand and use all these concepts. However the approach has been to consider a bottom-up approach: this means that a particular problem is considered first with the expert behaviour and concepts involved in addressing this problem. Concepts are added to the system's understanding as they become needed. They are classified below in an extrapolation of O'Keefe's view (1986b) about knowledge for advisory systems in simulation experimentation (see also section 2.5.5.ii).

Desirable concept understanding :
- description of the system
- entity and attribute
- activity
- state of the system and change in the state of the system
- endogenous or exogenous variables
- deterministic and stochastic components
- (questioning of) assumptions
- aggregation
- experimental design
- analysis
- statistical significance, confidence intervals
- congestion
- arrival patterns, inter-arrival times
- service process, service time and capacity availability (mean service rate, distributions)
- queueing discipline (LIFO, FIFO, random searching, baulking,polling)
- statistics (counts, summary measures, utilisations, occupancy, distribution, transit time)
- consistency
- autocorrelation
- steady states
- replication, batch means, regeneration

Figure 4.3 : Desirable Concepts for simulation experimentation

(a) Knowledge about Models

Knowledge about models may seen as description of the components in the model. Such components can be queues, entities, performance measures and parameters (Mize and Cox, 1968).

(b) Knowledge about Tasks

Such knowledge should include the objectives and general solution practices. Solution practices require strategies.

(c) Knowledge about Strategies

This involves understanding how strategies are specified. Strategies can be observed from "experts". Strategies involve the manipulation of parameter values and the use of different design procedures.

(d) Knowledge about Experimental frames

This is expertise about how to set up experiments that will yield significant results. It also entails knowledge about how to refine existing experiments. This knowledge deals with run-in times, run times, number of replications and variance reduction techniques, and may take the form of algorithms.

(e) Knowledge about Preliminary Analysis

This knowledge may be statistical in nature. It requires concepts such as means, variances, standard deviations, significance. It may be obtained from books as standard explicit knowledge.

(f) Knowledge about Evaluation criteria

There are a number of ways of evaluating performance. This knowledge may also be obtained from books in relation to formal statistical methods. But it may also be obtained from experts who may consider particular heuristic cost functions more appropriate in some cases.

(g) Knowledge about Analysis

This involves considerations of objective and subjective assessment of performance.

4.3.3 Formalisation

The links between the concepts and the hypothesis space must be formalised. It must resolved how the concepts to be represented are to be used by the system to present solutions. This step also involves designing a structure to organise the knowledge.

(i) Concepts and hypothesis space

A consultation process is to support the framework. The reasoning process is to be executed by the management of a number of rules. There are to be metarules to direct the search for appropriate rules at the level of task selection. Each task is to be supported by a set of primary

rules, which may be termed "strategic task" rules. To support these strategic rules, there will be a number of secondary rules that will hold knowledge that is only secondary to the strategy employed. A general subset of these secondary rules will be called design rules. Their responsibility is to ensure the correct design of experiments.

Strategies will give rise to ACTIONS. Actions will express the next step in the consultation process that is to take place. An action of type 0 quits the experimentation process and proceeds to the result presentation phase. An action of type 1 specifies the next experiment to be carried out. Finally an action of type 2 expresses a global strategy involving several prespecified experiments.

There will also be a number of evaluation criteria rules and analysis rules to take care of the analysis. Figure 4.4 suggests where rules will be needed.

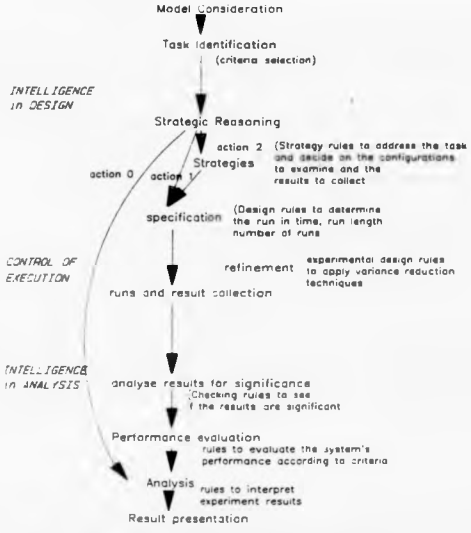


Figure 4.4 . Required rules for an advisor for experimentation

(ii) System Architecture

This stage involves designing a structure to organise the knowledge. The following formalisation was considered (figure 4.5).

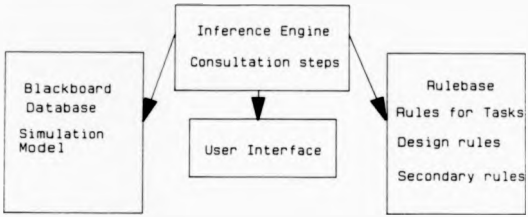


Figure 4.5 : Formalisation of the system architecture for the advisor

- Interpreter (an Inference Engine) is the centre-piece. It controls the consultation and has access to working memory, the user and the rule-base.
- The database (a Working Memory) stores all the facts specific to task and model under consideration.
- The user interface permits the advisor to obtain information from the user and provide advice to the user.
- Rule-Base :
  - Meta-knowledge : that controls which rules to consider depending on tasks.
  - Strategy task rules : general parameter manipulation
  - Secondary rules to support task rules in specification of experiment and the analysis of results.

The initial consideration is a production system whose knowledge is represented as a set of production rules ,i.e. [IF...THEN...] rules. The choice was prompted by experience that notes that production systems are appropriate for the following domains (Hayes-Roth, Waterman, Lenat

1983) :

1. Domains in which the knowledge is diffuse and ill-defined as opposed to those domains in which there are concise explicit models. This is the case in the domain of simulation experimental design.
2. Domains in which processes can be represented as a set of independent actions, as opposed to domains with dependent subprocesses. This is true in experimental design where each experiment can be represented separately from the next experiment.
3. Domains in which knowledge can easily be separated from the manner in which it is to be used, as opposed to cases in which representation and control are merged. Since intelligent input derived from manipulation of expert knowledge is to be of machine origin and the control is to be humanly performed, knowledge will have to be easily separable.

4.3.4 Implementation

As explained in chapter 2, the choice for implementation really comes down to using a shell or writing the system in a suitable language. At the time of this decision, three shells were available to the author for assessment purposes. They were found to be very restrictive in terms of parameter manipulations and in knowledge representation. In view of the research nature of the project, a shell was felt unsuitable because of the added limitations that it imposes.

Prolog was considered because of its acceptability as an AI language. It offers flexibility at the control level and further extensions of the program could be incorporated. Although it was decided to consider a production rule system, Prolog as an implementation language offered more flexibility in the knowledge representation if this should become required. LISP was also considered but appeared to require much effort in the development of suitable control environments as opposed to Prolog which has an automatic built-in theorem-prover. The balanced weighed further towards Prolog in the light of previous research in the school (see Flitman, 1986).

4.4 Implementation of a framework for an Advisor

4.4.1 Introduction

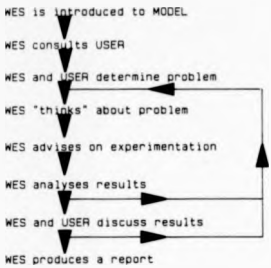
This section describes the implementation of an expert system framework for an advisor in simulation experimental design and analysis. The outcome will essentially be a shell that supports the control framework for a consultation. The actual knowledge and expertise regarding the tasks that such a shell could incorporate are discussed in the next section (4.5). The main control features of the system will be reviewed and the basic philosophy for using the system will be addressed.

The main assumption in using the system is still the existence of a verified and validated computer simulation model (eg. the die-shop model). The system has been called the Warwick Expert Simulator (WES) system. WES is concerned with offering advice on the design and analysis of experiments to address particular users requirements. These requirements are the tasks identified earlier (i.e. prediction, evaluation, comparison, sensitivity analysis, functional relations, transient behaviours and optimisation).

The structure of the system is divided into seven parts :

- setting up WES
- initialisation of consultation
- problem investigation
- problem solving
- problem solution execution
- discussion of results
- report generation.

This structure supports the following pattern of a consultation with WES, which mimics the behaviour of an expert :



With each of these steps, it is possible to associate a head predicate that controls the operations for each phase. Although Prolog is essentially considered a non-procedural programming language, it can easily be used to implement procedural patterns. The top level interpreter can thus be implemented:

```
prolog_engine :-
        meet_model,
        hello_user,
        needs,
        decide,
        controller,
        discus,
        report,!.
```

The diagram above suggests the need for possible re-entry mechanisms in the loop after successful completion of other steps. A better implementation became :

```
prolog_engine :-
        meet_model,
        hello_user,
        needs,
        solve,
        report,!.

solve :-
        decide,
        controller,
        discus,!.
```

Each step can be viewed as a separate module in its own right thus giving the program a modular nature which was implemented by characterising every predicate of a step with a prefix.

For example : in( ) for the introduction module, ps( ) for the problem solving module. This organisation of the program was made necessary by an early Prolog implementation used. Only a very limited amount of code could be held in memory at one time. Modules had to be moved in and out as required and the prefixing permitted easy retrieval and retraction. For example :

```
retractall(in(_)),!.
```

would retract all the predicates of the introduction module. Although a subsequent Prolog implementation no longer required these module manipulations, the author believes that this structuring greatly facilitated the maintenance and growth of the program.

4.4.2 Step 1 : WES is introduced to MODEL

The work of chapter 3 on the controller revealed that it is very useful, if not essential, to have a system that speaks in domain language rather than in generality. The same would apply even more in the context of an advisory system where it is important that the user understands what the advice refers to. For example, it appeared unacceptable to have advice in the form of :

set parameter_1 to value 2,

or Input utilisation of queue_2 :

General names are bound to lead to confusion and increase the risk of error.

To overcome this, it was decided to teach the system about the names of important components in the model. This takes place once. During this step, the user inputs the names of the queues, entities, seeds, performance measures and parameters & their default values. These names are used during consultation and are saved for further consultations.

The implementation of this module is through the head clause :

```
meet_model :-
        in(learn_names),
        in(save_model),!.
```

The predicate "in(learn_names)" deals with inputting names. This is done by five similar subclauses in(enames), in(qnames), in(pnames), in(pvnam), in(snames). Exceptionally, some detailed Prolog code is presented in this subsection. This is to offer a feeling about how control

mechanisms are dealt with. The WES technical documentation manual presents a detailed documented listing of the Advisory system code. Generally, only clause-names are presented in this chapter to identify the processes that take place. Knowledge representation mechanisms are however presented as they suggest how expertise may be handled.

Consider the task of finding out the names of the queues in the system

```
in(qnames) :-
        in(askname_q), answer(X),
        qnamenu(Number),
        asserta(qname(Number,[X],0,0,0)), nl, in(qcheck),!.

in(askname_q) :-
        write('Please input name of queue (stop if no more queue):'),
        read(X), retract(answer(_)), not(X == stop),
        asserta(answer(X)),!.
in(askname_q).

in(qcheck) :-
        qnamenu(X), qname(X,[Y],0,0,0), J is X+1,
        retract(qnamenu(X)), asserta(qnamenu(J)),
        in(qnames),!.
```

The clause in(askname_q) asks for the name of a queue. Provided the answer is not "stop", a fact gets recorded to designate the existence of the queue

```
qname(Number,[X],0,0,0)
```

The head "qname" means that the recorded fact is about a queue. 'Number' is the Prolog number for the queue called X. Apart from the expressed objective of having an advisory system that speaks in model-names rather than in general terms, the purpose of this representation is to give the advisory system some model knowledge. Although perhaps intuitively obvious, it was found that the level of support that may be offered by an advisory system in terms of further recommendations is limited by its understanding of the model. Therefore, in addition to the name, a number of attributes may be recorded. The three 0's therefore represent possible attribute values that are yet to be instantiated. They will be used to hold in the case of a service queue a lower bound of utilisation acceptability, an upper bound and a tag to identify whether this service queue is to be considered in this consultation.

The clause in(qcheck) checks that the new queue has been read into the database correctly and

updates the queue count (qnamenu(J)). It also calls the clause in(qnames) again in order to permit another queue name to be entered into the database. See appendix 13-b for an illustrative example of the implementation of this process.

When all the names have been entered, the clause in(save_model) will allow saving of the model described. This file can be consulted at the beginning of another consultation to save inputting the description of the model again.

4.4.3 Step 2 : WES consults USER

During this step, WES asks the user for general guidelines about the investigations that are to be considered. The user may choose between prediction, evaluation, comparison, optimisation, sensitivity analysis, functional relations and transient behaviour. (See figure 4.6).

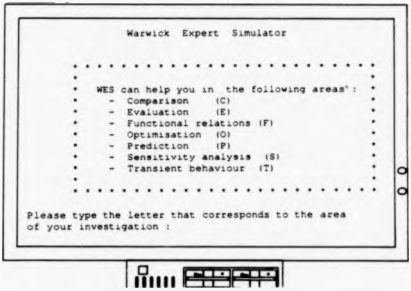


Figure 4.6 : screen offering choice of tasks to investigate.

The nature of the investigation is asserted as "problem([X])". This fact is used as early guidance for the rest of the consultation, which is carried out in step 3. A head clause "hello_user" also covers the essential tasks of introducing the system to the user (it(consultation)) and finding out the particulars of the user (it(user_details)).

4.4.4 Step 3 : WES and USER determine problem

Once the experiment type has been identified from step 2, WES asks questions specific to the nature of the problem. Answers from the user allow WES to build up a picture of the requirements of the user. These questions have been termed "consultation requests". They are directly related to the solution processes (rules) that have been implemented to address specific tasks. The idea is to obtain, in one go, as much information that may be relevant in the solution process. A discussion of the specific questions for each problem is reviewed in section 4.5. At the general level however, the user will often be required to express :

- the range on the parameter values,
- the criteria for evaluation purposes.

(i) Parameter ranges :

From questioning the user, WES obtains limits on the values of parameters. WES holds information about a parameter's default value, lower bound and upper bound in a fact of the form :

pname(Number,[Name],Default,Lower,Upper).

This is model knowledge as explained in section 4.4.2; it increases WES' understanding of the model. Parameter values during experimentation are held in facts of the form :

pstatus(Number,Experiment,Value).

(ii) Criteria for evaluation purposes

Correct evaluation practice requires that a criteria be established before the experiment is conducted. If only one configuration is considered, then normally the evaluation process would consider a number of quantification variables in turn and decide whether they are satisfactory. If two or more configurations are considered, then choosing the best one may involve some detailed multivariate analysis. Paired sampling may be tested formally using hypothesis testing. However in view of the difficulty this still poses, some analysts may be seen to compare configurations by looking at a global quantification value (Amer, 1982). Such criteria are often

objective orientated: for example minimisation of time in the system of entities. The approach adopted in this system is to compare configurations by looking at a global quantification value of the system's performance under these different configurations. Determining and comparing these values involves many heuristics about how to combine the different quantification variables of the model whose results are carefully derived.

WES has a number of criteria possibilities that use heuristics derived by the author. The difficulty in finding suitable heuristics is evident by the number of criteria that the system supports. WES holds the mechanisms for determining the evaluation criteria as module in its own right. A predicate "ecritenum" offers a choice to the user. (see figure 4.7)

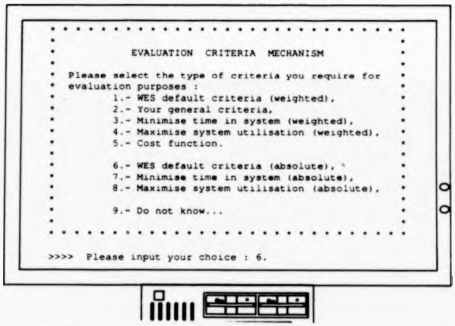


Figure 4.7 : screen presenting choice of evaluation criteria

1 & 6 WES default criteria (weighted and absolute)

These criteria take into account both the utilisation of service queues and the time-in-system of entities. It rewards high utilisation but penalises long time-in-system.

2 User general criteria

This is a simple modification of the default criteria. The user has the option of changing the bounds for good and poor performance for all or some of the queues.

3 & 7 Maximisation of service queue utilisation (weighted and absolute)

WES will consider the utilisation of queues. The criteria assumes that high utilisation equates with good performance. Lower bounds of good performance are set at 70% utilisation. Upper bounds for poor performance are set at 30% . A good utilisation receives a bonus point (+1), a poor utilisation incurs a demerit point (-1). During evaluation, WES simply totals the score for all or a subset of the queues in the system. The score is then assessed with the help of check_rules that take into account the number of queues in the system. This mechanism allows a user to select a subsystem of the model by naming only certain queues to be considered.

4 & 8 Minimisation of entity time in the system (weighted and absolute)

This implements a similar principle as the maximisation of service queue utilisations with the obvious modifications for a minimisation problem of time-in-system.

5 Cost function

A facility exists for inputing a cost function that will take as variables the parameters or the performance measures (user defined and queue utilisation) in the system. The user may then devise complex functions that are more suitable to the model under consideration. This facility was in fact rarely used.

4.4.5 Step 4 : WES 'thinks' about problem

At the end of step 3, WES should have a good idea of what the user wishes to find out about his model. During step 4, WES consults its rule-base. During the development stage of the advisory system, no probability factors were included in the rules. This approach is supported by Flitman (1986) who experienced that the incorporation of probability information into expect systems can slow rather than quicken expert response time. In chapter 7, however, the issue of

uncertainty factors is treated to develop more expert rules.

The search through its rule-base is guided in the first instance at a meta-level where the nature of the problem (eg. if it is a prediction problem) will determine the rules to consider. WES then matches up the requirements of the user with conditions in the rules. Figure 4.8 illustrates this process.

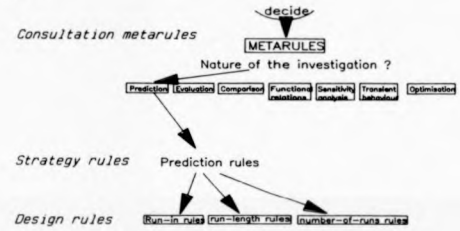


Figure 4.8 : Pattern of search through the rule-base

These rules may be regarded as first order rules (strategy rules) and discussion on their content is left to section 4.5, when each expert task will be reviewed. The approach is to pattern match the problem characteristics with facts in these rules in order to find a suitable solution procedure (strategy). Second order rules are then used to refine the experimental design (design rules). Such second order rules relate to run-in times, run length, number of replications.

For example, a prediction rule might be :

if the user wants to : - predict
- the utilisation of queue(JJ)
- over Y time units

then the following experiment should be carried out :
- run the model X times
- for Y time units
- with a run-in period of Z time units and
- monitor the utilisation of queue(JJ)

WES' thinking phase ends when a rule that applies has been found. Determining values for X and Z will require WES to call upon second order rules. The "expertise" of the rules is limited

to heuristics (see sub-module b in the WES technical documentation for the system's default rule-base) as the objective is only to demonstrate framework principles.

4.4.6 Step 5 : WES advises on experimentation

By the end of step 4, WES will have found a rule that will dictate the next action to be carried out. Usually, this will be in the form of setting up one or more simulation experiments. The user will be told (advised on) how to perform the next experiment, the number of runs, the duration of runs, their run in time, the configuration (parameter values) and the results to collect. The user is expected to input these results when the experiment has ended.

(i) Behaviour pattern

According to the task, there are a number of behaviour patterns that may be observed as WES leaves the strategic rules (see figure 4.9)

- action 0 : means no (more) experimentation and proceed with consultation.
- action 2 : means prepare to implement a strategy that involves the execution of several experiments.
- action 1 : means execute an experiment.

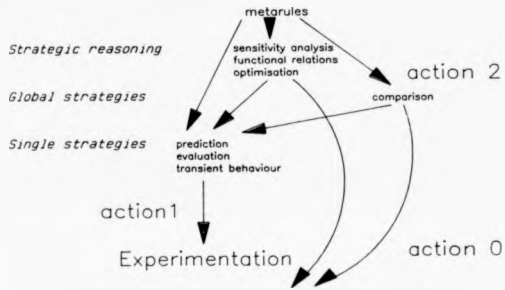


Figure 4.9 : implementation of experimentation behaviour patterns

To control these processes, there is a "controller" clause for each different type of experiment. All experiments involve running the model but they differ in the response expected from WES. For example :

Single strategies : involve the consideration of one configuration only. This occurs in the tasks of prediction, evaluation and transient behaviour. There is no return to the strategic reasoning level after the execution of an experiment.

Strategic reasoning : begins with the design of a starting experiment. After the execution of this experiment, the controller returns to the strategic reasoning level where further rules are considered. Two possibilities occur: either the rules suggest that another experiment should be conducted and the process will be repeated, or it is decided that enough experimentation has taken place and the consultation process should proceed. This pattern is used for sensitivity analysis, functional relations and optimisation task investigations.

Global strategies : involves the planning of several experiments prior to any execution. This is used in comparison exercises. After the execution of an experiment, the controller returns to examine rules to determine whether more experiments have been planned or whether the global strategy has been completed and the consultation should proceed.

(ii) Experiment Specification

The actual specification of experiments is guided by the clause 'frame_1'. At this stage, the implementation presented only considers one type of experimental approach. However, the system supports a structure that allows the inclusion of any number of expert experimentation processes. Illustration of this will be presented in chapter 5.

An example of experiment specification is given in figure 4.10. It considers an experiment with the die-shop model.

This example demonstrates how the original goal of this chapter is achieved. The experimentation process was to have machine intelligence and human control of execution. WES has decided on the experiment. The simulation-user must carry it out.

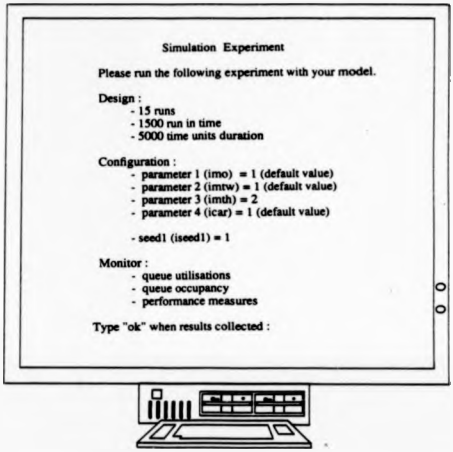


Figure 4.10 : screen informing user of experiment in progress

(iii) Results collection and analysis

Results will be passed to WES through the clause "results". Each result will be collected and stored it in a fact:

wfac(900,Z,[X],Ha,Answer)

where 900 is the fact characterisation,

Z is the experiment number,

X is the name of quantification variable (either a service-queue or entity),

Ha is the run-number of that experiment

Answer is the value of this quantification variable during that run.

The clauses ps(procqu), ps(procen) and ps(procpv) process these individual results. For each

quantification variable, the mean is computed for its average utilisation for the entire experiment, together with a standard deviation for this value in order to establish upper and lower bounds. Aggregate results are stored in facts of the form (for example for queue utilisations):

qustatus(Qnumber,Exp,Mean,Up,Lo,0)

where Qnumber is the number assigned to the queue by WES,

Exp is the experiment number,

Mean is the mean value,

Up is the upper bound on the mean,

Lo is the lower bound on the mean.

When bounds are calculated, a clause ps(check_t_value) checks heuristically the significance of the confidence interval. Prolog permits the storage of a t-table database. If the interval is too large, this fact is recorded.

4.4.7 Step 6 : WES and USER discuss results

The first stage of this step is taken up by WES analysing and checking the processed results for significance. If the variance of mean values is too large, WES consults its rule base on what to do and attempts some variance reduction technique. Step 5 is then executed again and the user is told how to extend the experiment he has just completed in order to obtain significant results. Step 6 will be attempted again.

Once WES is satisfied that the results collected are significant, it will evaluate the performance of the system during the experiment using the criteria selected. It will then present its findings to the user. The user may then either return to step 2 for further experimentation and investigation of the simulation model, or proceed to step 7.

(i) Experiment Refinement

WES checks to see if a fact marking the failure of significance has been asserted. If so, experimental rules are used. A set of heuristics are implemented. Generally, they consider the number of observations obtained. These will be increased up to 30. For example:

If - using batch runs,
  - twice the number of runs is less than 30,
Then - extend old experiment with twice number of runs.

Should the system no longer find a rule to use to refine the experiment, the system interrupts (see appendix 13-a for illustration) and requests help from the user who may:

- increase the number of runs past 30,

- restart experiment with longer run in period,

- restart experiment with longer length of runs,

- tell WES to ignore lack of significance and proceed with consultation pattern.

(ii) Performance evaluation

Performance is found through clauses ar(find_performance). There are several such clauses: one for each of the different evaluation criteria. They are used to derive a single quantification assessment of the performance of the system under the configuration used. The assessment is stored in fact *wfact(100,S,A,[X])*

where 100 denotes the nature of the fact (ie experiment performance),

S is experiment number,

A is performance quantification values,

X is performance qualification string.

(iii) Presentation of results

The results are presented using predicate "discus". There is a different presentation of results for each expert task. Generally, it involves the performance of the system under the different

configurations used. Further discussion is kept to section 4.5 when the implementation of the expert tasks is reviewed.

(iv) Discussion

The clause "continue_consultation" permits the user to choose the course of action (see figure 4.11)

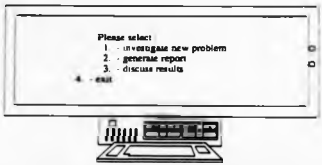


Figure 4.11 : screen offering choice for course of action

Discussion of results is limited to WES offering some general advice derived from the results presented. Analysis rules are used. In fact this facility was not developed to any great level of expertise, as a major limitation was the system's understanding of the model structure. Consideration of this problem is given in section 7.3.

4.4.8 Step 7 : WES produces report

This stage produces a report that summarises the consultation, presents the experiments performed and the findings of the investigation (see appendix 6-b for an example). There is nothing new in the report that the user has not seen during his interactions with WES. The clause "report" directs the generation of the report. Again the discussion relating to the material presented in the report is left to section 4.5 that covers the implementation of these tasks.

At the same time as producing a report file, the clause rg(savecon) produces a file that stores in Prolog code all the findings of the consultation (wfac, qustatus, etc.). The file if it exists can be used for further consultations. A point that caused great concern was to determine the

level of granularity required in the storage of results. Initially, only aggregate results were stored. For example, average utilisation of service queue X is 90% was stored as opposed to the individual results on which this result is based. However, as the rules were refined, it was found to be more useful to have the individual results. This was mainly due to the fact that aggregate results are only computed for the quantification variables which the user expressed as wishing to see. Individual results are recorded for all variables. The Prolog database suitability made these results easy to reconsult. See appendix 6-a for an example of such a file.

Finally, the clause more_consultation offers the user the chance to pursue their consultation with WES (see figure 4.12).

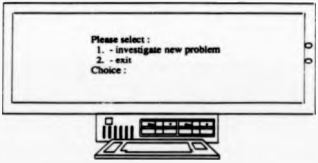


Figure 4.12 : screen offering choice of course of action after report generation

4.4.9 Conclusion

The section has presented an implementation structure for an advisor. The system advises on the design of experiments and assists in the interpretation of results. The advisor adopts a consultation pattern and the intelligent process supports strategic reasoning that allows the system to react to results and prepare further experimentation. Only heuristics have been used as system knowledge and the evaluation procedure has been simplified to the consideration of a general quantification value only. The structure should be general enough to allow expertise to be added concerning any expert task that is to be investigated. This is illustrated in the next section.

4.5 Expert Tasks for a Prototype Advisor

4.5.1 Introduction

In section 4.4, the feasibility to implement the structure of a system that could support a consultation framework was established. It was seen how potentially the system could interrogate the user to discover his requirements, "think" about the problem, design experiments to be conducted by the user, analyse the output from these experiments and interpret them before presenting them to the user with the option of generating a report of the whole investigation.

In this section, the emphasis is on discovering what tasks can actually be addressed using this framework. As mentioned in the previous section, the initial part of the consultation asks the user to decide which task to address. While developing WES, a prototype approach was adopted. In the first prototype only the task of prediction was addressed. Once this prototype was working to a satisfactory level, work began on the next task of evaluation. With each new prototype, the capability of carrying out an extra expert task was implemented. The objective is to implement a default strategy to support these tasks and investigate the amount of support that may be provided.

The following description shows how WES actually carries out each of these expert tasks. The description is chronological with reference to the implementation of each task. The task objective is explained; the consultation requests, the default strategy rules and the output are presented and an example is provided. Additionally, appendix 13 contains example consultations using all these tasks.

4.5.2 Prediction

(i) Task Objective

Prediction was taken to be the evaluation of a mean value for one or more given variables in the model. For example, the user may ask for the mean utilisation of a service queue, or average

time in the model of some entity type. Alternatively, the user may ask for the mean value for some user defined performance measure.

The prediction task was implemented first as it was felt to be the most fundamental in the simulation experimental design phase. It has the lowest level strategy. All other simulation expert tasks were expected to make use of prediction. The prediction task retrieves results by setting up experiments using design rules. It does not address the problems of parameter manipulation. The other tasks would normally do this and call upon prediction rules to set up the experiments. See appendix 13-a for a consultation example using the prediction task.

(ii) Consultation Requests

In a prediction investigation, WES will ask the user for the some guidance about :

- the variables that are to be predicted,
- the configuration,
- the time span for the predictions.

a. variables

In the system's final version, the user is offered a choice of prediction:

- utilisation of queues,
- time-in-the-system of entities,
- occupancy of queues,
- user-defined performance variables,
- combination of the above.

Initial work only focussed on queue utilisations and user-defined performance variables that are counts. A subclause will retrieve from the user the identity of all the variables that are to be monitored. This approach may be likened to the notion of selective instrumentation where only data relevant to a particular goal is gathered and analysed (Reddy & al, 1985).

b. configuration

The user may change some of the parameter values from their default values. Changed parameter values are held in facts of the form

pstatus(Number,Experiment,Value)

where Number is the parameter number as assigned by WES,

Experiment is the experiment number,

Value is the parameter value for the next experiment.

c. time

The clause pi(predtime) asks the user if he knows what time value is to be considered. For example, if the user wishes to consider the operation of the die-shop plant over one week, then he has the option of inputing this duration. If he does not care, the problem will be treated as a non-terminating system and WES will decide on appropriate lengths of runs.

If the user does know the time span that is to be considered, he is asked whether the system is terminating. If the answer is 'yes', then no run-in time is considered. If the answer is 'no', WES will need to find an appropriate run-in time. A number of rules are implemented to this effect.

(iii) Default Strategy Rules

If the problem is a prediction investigation, WES will be directed to "pred_rule_(X)" type rules. They all have the same structure. They are to be read as production rules :

if : the following conditions hold
then : take action

The conditions are :

frame(Xf) : use experimental frame type Xf
barun(1) : experimentation using batch runs
serun(1) : experimentation using sequential runs
bsrun(1) : experimentation using batch runs followed by sequential runs
quexp(_,_) : look at queue utilisation ?
poexp(_,_) : look at queue occupancy ?
pvexp(_,_) : look at performance variables ?
configuration : configuration can be found
run_in_rule_(Xa) : run-in time can be found
run_time_rule_(Xb) : run time can be found
run_num_rule_(Xc) : number of runs can be established

Then the possible action to take is :

action(A,B,C,D,E,F,G,H,I)

where

A : option is a frame selection,
B : experiment number
C : run in time
D : length of run
E : number of first run
F : number of last run
G : monitor queue utilisation if G = 1
H : monitor queue occupancy if H = 1
I : monitor performance measures if G = 1

As can be seen, a number of secondary rules support a prediction exercise :

- run_in_rule_(Xa)

- run_time_rule_(Xb)

- run_num_rule_(Xc)

An example :

```
pred_rule_(1) :-
        quexp(_,_),
        qoexp(_,_,_,_),
        pvexp(_,_),
        configuration,
        frame_(1),
        bsrun(1),
        run_in_rule_(Xfind),run_in_time_is(X),
        run_time_rule_(Yfind),run_length_is_(Y),
        run_num_rule_(Zfind),ifirstrun(Za),ilastrun(Zb),
        expnumber(E),
        assert(action(1,E,X,Y,Za,Zb,1,1,1))
```

which has the translation :

pred_rule_(1) applies

if - queue utilisation predictions are required [quexp(_,_)]

- queue occupancy predictions are required [qoexp(_,_,_,_)]
- performance measures predictions are required [pvexp(_,_)]
- a configuration can de determined [configuration]
- experimental frame of type 1 should be used [frame_(1)]
- results are to be collected using batch runs followed by sequential runs [bsrun(1)]
- a run in time can be found [irunin(X)] using run-in rules [run_in_rule_(Xfind)].
- a run time can be found [itime(Y)] using run time rules [run_time_rule_(Yfind)].
- the number of the last run completed [ilastrun(Za)] and the number of the last run to complete [irun(Zb)] can be found using number of run rules [run_num_rule_(Zfind)].

then take action (1,E,X,Y,Za,Zb,,1,1,1) where :

- 1 refers to the type of experiment (here 1 means run the model)
- E is the experiment number
- X is the run in time for runs
- Y is duration of runs
- Za is number of first run to be done
- Zb is number of last run to be done
- 1 means monitor queue utilisation
- 1 means monitor queue occupancy
- 1 means monitor performance measure

Once an applicable rule has been found, it is fired. (See appendix 5 for a description of such operations). Results are collected and tested for significance as described in section 4.4. If results are not significant, WES will consult in its rule-base, rules of the form exp_design_rule_(Y). These rules deal with improving an existing experiment so that it will yield significant results on which decisions can be made.

(iv) Output

The output to the screen and to the report file are essentially the same. Two clauses, dr(lookmore) and rg(repinto) cover these procedures respectively. For a prediction exercise, a typical example of output from firing the above rule might be as in figure 4.13

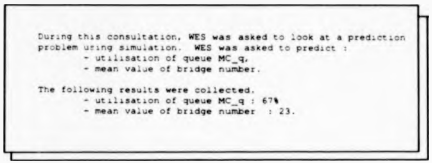


Figure 4.13 : WES report file from a prediction consultation

All these results are retrieved from facts of the form :

qustatus(Qnumber,Expnumber,Mean,Lob,Upb,0)

4.5.3 Evaluation

(i) Task Objective

Evaluation was the second expert task to be implemented. The objective was to develop a facility that would interpret results obtained from predictions. These first two expert tasks of evaluation and prediction were expected to complement each other, to provide the skeleton of a system that could help with the experimental design and analysis of results of experiments.

Evaluation was defined to be the assessment of performance of a system against some specific criteria. Once the user has specified the kind of evaluation required with the appropriate criteria (from the module described earlier), WES consults its rule-base to decide on an appropriate strategy. This strategy is likely to be in the form of running a prediction experiment.

The outcome produces an appraisal of the system being evaluated. It highlights parts that are working well and points out troublesome areas.

(ii) Consultation Requests

The consultation request are limited to the configuration to be evaluated and the criteria to be used. Initially, the module for evaluation criteria was included in this section. The complexity and a concern to keep the system modular eventually required a separate module.

(iii) Default Strategy Rules

Experimental design rules for evaluation purposes have the following possible conditions :

- has the criteria been established ?
- does WES know what is to be monitored ?
- has a prediction rule been found applicable ?

The result of affirmative answers will cause the rule to be fired. This will mean that a suitable strategy rule has been found for the task in question. Also note how the prediction rules will take care of setting up experiments.

(iv) Output

In section 4.4, it was explained how aggregate results of the experiment are stored in a fact with characterisation 100 holding a quantitative and qualitative assessment of the systems performance. For example : the fact

wfac(100,5,105,['This system performs well'])

may have been stored. Initially, the qualitative assessment was presented. Much work was spent on finding appropriate qualitative statements and matching them to the quantification value. But this proved more difficult than originally anticipated as each model offers a different range on its quantification values. For example, some models vary in quantification performance between 98 and 102, whilst others may vary from 101 to 110. In one case, the value 102 should be accompanied by the qualitative "very good" and in the other with "poor". The global quantification value is appropriate as a relative guide in comparing several configurations but offers no real insight into the performance of a single configuration.

Eventually, further refinement was abandoned on the qualitative statement approach and enhanced by a presentation of the values of the different quantification variables. It was felt that this offered a simulation user a suitable environment to gain his own appreciation of the model's performance in the light of representative values. The user was however still offered support by WES: in a separate section, WES points out the values of quantification variables that are outside the bounds of acceptability. These boundary values may be either declared or default ones. Default values exist only for service queue utilisation. Below 30% and above 70%, it was felt that the user be made aware of these facts. These values may be changed as may bounds be declared for time-in-the-system of entities or other performance measures.

(v) Example

The die-shop was evaluated using WES. The default configuration was chosen using WES default evaluation criteria. The report file reads as in figure 4.14.

```
During this consultation, WES was asked to evaluate the performance
of the whole system using :
  - WES default evaluation criteria mechanism

The following experiment was conducted :
  - the model was run 7 times for a period of 5000 time units after
    a run in period of 2000 time units.

The conclusion that WES came to was that the system performs well
(score 105).

The following results were collected:
        - utilisation of service queue MC_q : 98%
        - utilisation of service queue VEDM_q : 70%
        - utilisation of service queue WEDM_q : 45%

        - mean value of bridge number: 34
        - mean value of plate number: 36
        - mean value of back number: 37

Please note:
        - the high utilisation of queue MC_q :98%
```

Figure 4.14 : WES report file from evaluation consultation

See appendix 13-a for a complete consultation using the evaluation task.

4.5.4 Comparison

(i) Task Objective

This facility involves investigating the merits of different system configurations or different operating policies and procedures. Once WES has identified what is to be compared, its rule-base will decide on an 'action' that will combine the use of the prediction and evaluation mechanisms. The comparison task offered the chance to implement a low level impression of strategic reasoning. The approach is to devise a global strategy that will involve comparing and evaluating each configuration and operating policy and procedure with each other. The criteria for deciding which is better is set up by consulting the user. Appendix 13-b provides an example consultation using this task.

(ii) Consultation requests

WES will need to know :

- how many configurations are to be compared;
- a description of each configuration in terms of parameter values;
- the evaluation criteria to be used.

(iii) Default Strategy rules

Initially there were only three rules controlling the comparison task.

```
(a) comp_rule_(1) :
        if  - comparison experimentation has not yet begun
            - there is a number of configurations to compare
        then - find an evaluation rule to use on first configuration
             - assert global strategy (action of type 2).

(b) comp_rule_(2) :
        if : all configurations have been compared
        then leave experimentation (action of type 0)
```

(c) comp_rule_(3) :
if : comparison experimentation has begun
then : get next configuration
update progress of global strategy (action of type 2)
find new experiment (action of type 1).

In essence, the comparison rules control the number of different configuration experimentations that are to be carried out. The experimentation on each individual configuration is carried out using evaluation rules which in turn use prediction rules.

(iv) Output

The evaluation output mechanism as explained in section 4.4 provides a quantitative and qualitative assessment of the system's performance that is stored in a fact of the form

wfac(100,Expnumber,Value,['Comment']).

When several configurations are compared a clause dr(find_best) finds the best performing configuration and stores this in a fact of the form

wfac(200,Expnumber,Value,['Best performance']).

This permits easy presentation of results.

(v) Example

Three different configurations of the die-shop model have been compared by WES. Below in figure 4.15 is the output to the report file.

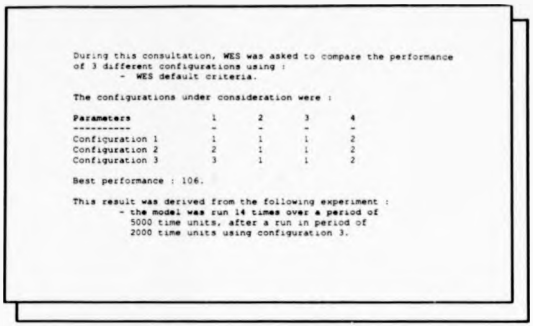


Figure 4.15 : WES report file from comparison consultation

4.5.5 Sensitivity Analysis

(i) Task Objective

This expert task extends the previous task of comparison. Once a general criterion (be it a cost function) has been identified as a means of assessing performance of a system, WES can carry out some sensitivity analysis. The task of sensitivity analysis is limited to varying all the parameters by a small amount and note the change in performance. This task offers the chance of implementing the notion of strategic reasoning, where the system reacts to results from previous experiments. The main idea is to start with a default configuration and change each parameter in turn by a small positive increase and a small negative increase. Experiments are run in order to decide which of many factors are most significant in affecting overall system performance.

(ii) Consultation Requests

The clause pi(sensparam) will obtain from the user the bounds on the parameter values. This is necessary to prevent WES from attempting to run experiments with parameter values that

are not acceptable. And the clause "ecriterium" finds the evaluation criterion.

(iii) Default Strategy rules

WES holds a number of different rules in its rule base regarding sensitivity analysis procedures. The first rule is a check.

```
sens_rule_(1) :-
        if no parameters in the model
        then quit experimentation (action of type 0).

sens_rule_(2) :-
        if - parameters can be changed
           - but no experiment has been run
        then run the model with default values.
```

The rest of the rules are concerned with manipulating parameter changes between experiments. Some rules exist to return to the original configuration before the next parameter change. To keep track of the last parameter change, a fact :

```
wfac(400,Experiment,Parameter,Value,Increase)
```

is asserted, where

400 is fact characterisation,

Experiment is experiment number,

Parameter is number of parameters being changed

Value is new value

Increase : 1 means positive increase, 0 means negative increase.

An example rule for parameter change is

```
sens_rule_(6) :-
        if - parameters can be changed
           - find current parameter under investigation
           - last change was +1 on that parameter value
           - can go -1 on default value of that parameter
        then run the model with parameter value = default value -1
```

The code for this rule is :

```
sens_rule_(6) :-
        expnumber(Z), V is Z-1,
        sensparnumb(X), Y is X+1,
        pname(Y,[r],S,T,U), pstatus(Y,V,D),
        wfac(400,V,Y,D,1), S > T, Da is D-2,
        not(wfac(400,_,Y,Da,0)), retractall(pstatus(Y,Z,Da)),
        assert(wfac(400,Z,Y,Da,0)),
        eval_rule_(Xx),!.
```

The process of control in the sensitivity analysis task is for the sens_rules to determine the required configurations and then to use eval_rules and in turn pred_rules to evaluate these configurations.

(iv) Output

Once the experimentation is completed, WES will analyse results. In the context of sensitivity analysis, WES will find the contribution to performance of every change made (dr(contribution)). Those changes are stored in facts

wfac(420,Experiment,Performance,Change,Increase).

Then WES works out :

the best change (dr(find_bigger),

the worst change (dr(find_smaller)),

presents the best results (dr(sens_pos_results)),

presents the worst results (dr(sens_neg_results)),

and finally list all results (dr(sens_all_results) if requested.

(v) Example

Sensitivity analysis was carried out on the die-shop model. Output to the screen is seen in figure 4.16. Appendix 13-d illustrates a complete sensitivity analysis consultation.

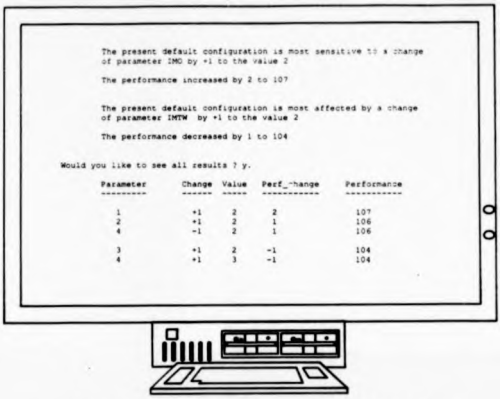


Figure 4.16 : screen output for sensitivity analysis investigation

4.5.6 Functional Relations

(i) Task Objective

Having developed a tool to perform some sensitivity analysis on the model that would present parameters offering the best response rate when subject to change, it was seen natural to consider a tool that might examine the nature of relationships between one or more significant factors and the system's response. The early objective was simply to carry out some linear regression on the values of given parameters against the model's overall performance measure as defined by the user, and present a linear relationship to the user.

(ii) Consultation Requests

WES will simply ask the user for the name of the parameters that are to be considered, and

their range of parameter values. WES will also ask for the evaluation criteria to be used.

(iii) Default Strategy rules

To perform this expert task, a number of strategic functional relations rules exist supported by a number of secondary rules. The strategy rules have "f_rel_rules_(X)" type. They are responsible for the main operations in the execution of the task. For instance, they select a parameter of interest, vary its value and apply evaluation rules for each different configuration. An example rule is :

```
f_rel_rule_(4) :-
  if    - looking at a parameter
        - new change would not be possible with current parameter
  then  - find a new parameter to change
        - set values back to default
        - check for range
        - find this new configuration
        - check to see if this experiment done before and
              - either find eval_rule and proceed,
              - or assert old results for this parameter and find
                new change.
```

The secondary rules that support the "f_rel_rules" are "f_range_rules". These rules are used to determine the step in a parameter value from one experiment to another. The purpose is to limit the number of experiments that are to be carried out. An example is :

```
f_range_rule_(1) :-
  if    - there is a parameter to look at
        - the lower bound is not equal to upper bound
        - the range from lower to upper is less than 5
  then  - assert the step for investigation is 1
```

(iv) Output

As a first attempt, the data from all the experiments were collected and stored in facts of the form

wfac(500,Conf,Exp,Pnumber,Value,Performance)

where 500 is fact characterisation

Conf is configuration number

Exp is experiment number

Pnumber is parameter number as assigned by WES

Value is parameter value for that experiment

Performance is quantitative value of the system

These results were then processed by Prolog clauses that performed some linear regression on the data. This had limited success in view of the difficulty of numerical calculations with the Prolog available at the time. In fact, it was felt more helpful to present the data to the user for him to investigate the relationship, since most relationships are not linear in any case.

(v) Example

The die-shop model was not really suitable for functional relations investigations. The range of the parameters were limited from 1 to 3. However, by requesting to see the effects of changes in values of parameters IMO and IMTH, the user could grasp some important feel for the relationship of these parameters to performance. Below an example of the output is given in figure 4.17.

Figure 4.17 : screen output from a functional relations investigation

In this example, the user can see the effect of the different parameter changes without the need of regression analysis. Appendix 13-e presents a consultation using this task.

4.5.7 Transient Behaviour

(i) Task Objective

A weakness that became apparent was that the system worked on time limits rather than conditional bounds. Ie. experiments were set up to start and stop according to the simulation clock rather than according to events or occurrences in the system. It appeared important to complement the tasks of evaluation and prediction with the ability to look for specified transient behaviour such as bottlenecks, excessive queue buildups or utilisation imbalances. A request for the use of this task provides a report of these transient behaviours.

(ii) Consultation Requests

WES will need to know:

- what is to be monitored
- when to start monitoring
- when to stop monitoring

The choice of factors to monitor is the same as for the prediction experimentation. Monitoring conditions were of two kinds :

- when clock reached specified time
- when given factors became >= a given value

(iii) Default Strategy rules

The structure of the rules that support the transient behaviour task have simple form :

- find out if queues are to be monitored
- find out if performance measures are to be monitored
- determine the starting conditions
- determine the stopping conditions

These subtasks are executed by a series of secondary rules.

The idea is to use these secondary rules to set up the appropriate monitoring environment before consulting a prediction rule to set up an experiment in the light of this environment. The monitoring environment is created by the assertion of a series of action 5 facts.

Eg. action (5,St,[A],[B])

where 5 is characterisation of the action type

St is binary : 1 = start condition , 2 = stop condition

A is name of factor (can be time)

B is critical value

Somehow this environment has to be made present in the experimentation process. The user has to be told about these conditions. To achieve this an extra clause was introduced to the predicate "frame_1" that presents the experiments to the user.

```
frame_1 :- .
            .
            ps(checkaction(5)),
            .
            .
            .
ps(checkaction(5)) :-
        action(5),
        ps(check_start_(X)),
        ps(check_stop_(Y)),!.

ps(check_start_(1)) :-
        write('Monitoring to start when '),nl,
        ps(wr_start_(X)),!.

ps(check_stop_(1)) :-
        write('Monitoring to stop when '),nl,
        ps(wr_stop_(Y)),!.
```

The clauses ps(wr_start_(X)) and ps(wr_stop_(Y)) will present the conditions as stored in facts action(5,_,_,_).

(iv) Output

The output will present the conditions in which the experiment was carried out and give an analysis of the results in the format of prediction experimentation output.

(v) Example

WES was asked to consider the die-shop model and produce a report on the transient behaviour of queue IMAQ(11) between time 500 and the moment performance measure IBD reached 4. The results were as presented in figure 4.18.

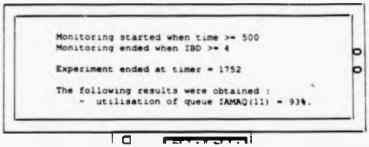


Figure 4.18 : screen output from a transient behaviour exercise

See appendix 13-c for illustrative consultation.

4.5.8 Optimisation

(i) Task Objective

This last expert task is perhaps the ultimate goal of any simulation exercise (Fishman, 1973). This phase calls on all the other expert tasks. The first step is to define an evaluation criteria. The next step is for WES to discover the limits on the parameters. The approach thereafter is to use step-wise non-linear optimisation methods guided by the advisory system and the user in order to approach an optimal configuration for the criteria used.

(ii) Consultation Requests

The consultation requests are dealt with easily by two existing clauses :

- "ecriterium" helps to extract from the user the evaluation criteria to be used.
- pi(sens_param) obtains the bounds on the parameter values.

(iii) Default Strategy rules

The initial set of rules were a basic set of non-linear optimising rules. The idea was to start from the default configuration and vary one parameter by positive or negative amounts and see if there was an improvement in the system's performance. If yes, change the parameter again by same positive increase or same negative amount and continue in same direction so long as change is improving the performance. When change no longer improves performance, assert previous "best" configuration as starting configuration and repeat the process with new parameter. Eventually, the process will converge on a "best" configuration for the evaluation criteria chosen. (See appendix 13-f for an optimisation consultation that uses a different set of rules).

A fairly large number of rules are required to support this task. Also, a certain number of new facts need to be devised to keep track of current status of investigation. Consider :

wfac(300,Conf,P_Number,Value,Dir_change),

where300 is the characterisation of fact

Conf is the configuration (experiment) number

P_number is the parameter number currently being changed

Value is the current value of this parameter

Dir_change is binary : 0 means negative increase, 1 means positive increase.

Also used are the facts devised earlier :

wfac(100,X,U,Z) stores performance of an experiment

wfac(200,R,S,['Best performance']) stores the best configuration

Different types of rules are used.

optim_rule_(1) :- checks that experimentation is possible (ie parameters can be changed).

optim_rule_(2) :- updates record of best configuration so far.

optim_rule_(3) :- keeps track of present starting configuration.

optim_rule_(4) :- starts off all optimisation exercise by running default configuration.

optim_rule_(X) :- all the other rules deal with changing the parameters according to experimental results so far, and preparing new experiments using eval_rules. For example:

```
optim_rule_(9) :-
        if    - parameters can be changed
              - last experiment was best
              - last change was -1
        then  -       go -1 again if possible
```

which has code :

```
optim_rule_(9) :-
        pnamenu(Y), Y > 1,
        expnumber(K), X is K-1
        wfac(100,X,U,Z),
        wfac(200,R,s,['Best performance']), X == R,
        wfac(300,X,A,Ds,B), B == 0,
        pname(A,[E],F,G,H),
        pstatus(A,X,C),  G < C,  D is C-1,
        retractall(wfac(320,_,_,_,_,_)),
        assert(wfac(320,R,K,A,D,0)),
        wfcheck_rule_(1), not(wf_true(320)),
        assert(pstatus(A,K,D)),
        assert(wfac(300,K,A,D,0))m
        wfcheck_rule_(2),
        not(ofail(1));(ofail(1),retractall(pstatus(A,R,_))),
        not(ofail(1));(ofail(1),retractall(ofail(1))),
        eval_rule(Xy),
        assert(arule(o,9)),!.
```

Supporting these rules are wfcheck_rule_(X). These rules check that the proposed experiment has not been run already.

(iv) Output

The report file of an optimisation exercise will present :

- a description of the problem [rg(repinto)]
- a reminder of the evaluation criterion used [rg(repcm)]
- a presentation of its final conclusion [rg(obest)]
- and if requested a description of all the experiments carried out [rg(odetail)].

(v) Example

The die-shop model was presented for optimisation purposes using WES default criteria. The report file read as in figure 4.19.

```
During this consultation, WES was asked to optimise the die-shop
model.  Performance was assessed using :
        -  maximisation of queue utilisation criteria.

The best performance achieved was 111 using configuration
        Parameter 1 (IMO) = 3
        Parameter 2 (IMTW) = 3
        Parameter 3 (IMTH) = 2
        Parameter 4 (ICAR) = 2


Details of the investigation
----------------------------

Ex_number       Perf    IMO     IMTW    IMTH    ICAR    Changed
---------       ----    ---     ----    ----    ----    -------
   1            105     1       1       1       2
   2            107     2       1       1       2       +2
   3            108     3       1       1       2       +3
   4            109     3       2       1       2       +4
   5            110     3       3       1       2       +5
   6            111     3       3       2       2       +6
   7            109     3       3       3       2       +4
   8            108     3       3       2       3       +3
   9            106     3       3       2       1       +1
```

Figure 4.19 : WES report file from an optimisation investigation

4.6 Conclusion

The concept of an advisor has been considered in this chapter. The aim has been to support a user from the intelligent component in experimental design and analysis. An intelligent reasoning model was put forward to understand the processes involved. The processes were seen to involve intelligence in design and intelligence in analysis. The design of experiments is perceived as a result of the application of strategic reasoning using strategy rules and design rules. The analysis of results occurs at different levels to check experiments for significance, evaluate a given experiment and analyse the results of the whole investigation.

By applying expert systems techniques, using the consultation paradigm, this model of intelligent reasoning was implemented in Prolog. The consultation pattern goes through the steps of :

- introduction to MODEL
- consultation with USER
- determination of problem
- "thinking" about the problem
- advising on experimentation
- discussion and analysis of results
- report generation.

A number of simulation experimentation tasks were addressed using this framework. Default strategies were implemented. Later, it will be considered how the knowledge of the system can be improved. The tasks that can be addressed are those of prediction, evaluation, comparison, sensitivity analysis, functional relations, transient behaviour and optimisation.

The framework proposed supports adequately this consultation process. The rules used are simple but provide evidence that advice from machine intelligence can assist the simulation-user in the design and analysis of simulation experimentation.

In the next chapter, it will be considered how this advisor may be linked up to the controller of chapter 3 in order to provide an integrated system that supports the user in both intelligent reasoning and control of execution.

End of Chapter

Chapter 5

Towards an Integrated Expert - Machine Intelligence & Control

5.1 Introduction

The motivation behind this thesis has been a desire to find ways of supporting the experimental design and analysis phase for a user in the context of a simulation study. The starting point has been the postulate that the process requires intelligent (non-procedural) reasoning and formal (procedural) control of execution. In chapter 3, it was argued that the quality of experimental design could be enhanced by the development of a machine based controller that formalised the execution process. Input was obtained from the user at the appropriate time. In effect, the machine did what the human intelligence told it to do.

In chapter 4, the alternative view was considered : the quality of simulation experimentation may be enhanced if the user is told how to control his experiments. An advisory system was developed using a consultation framework that supported the investigation into a number of simulation tasks. The outcome was that the human did what machine intelligence told him to do.

In this chapter, the objective is to combine the user support, derived in the previous two chapters by linking the intelligent controller to the advisor in order to achieve an integrated system that has machine intelligence and machine control. The consequence of such a system would be that the user is removed from the whole process of reasoning about and controlling the experimentation. A non-specialist user may proceed with confidence in the use of his model, by simply specifying the task to be investigated. In section 5.2, the existing controller and advisor are linked in parallel, and the notion of procedural control to non-procedural reasoning steps is investigated. Section 5.3 investigates sequential linking as a more practical implementation, and batch file programming is proposed as a suitable environment. Finally, section 5.4 considers necessary features in order to refine the integrated system into an expert for the design and analysis of simulation experiments.

5.2 Parallel Linking of the Intelligent Controller to the Advisor

5.2.1 Review of the Controller and the Advisor

The work of chapter 3 proposed a framework for experimental frames to be executed under machine control and specified by human input. In the latter part of that chapter in section 3.4, work was carried out to link a Prolog program to this controller thereby causing remote input into the execution of experimental frames. In the final section 3.5, the focus of interpretation changed in so far as the controller became intelligent with remote execution. Execution of frames originate from low level PROLOG-implemented intelligence. A link to the advisor may be viewed as making the controller more intelligent, through the use of a more sophisticated program : ie the advisor.

The advisor is PROLOG-based. It supports an entire consultation process, requiring input of different kinds. There is input to be obtained from the user about the nature of the investigation and there is input to be obtained from the model via the user interface about its characteristics and status. It is this second source of input that is of interest to the link. Such input is obtained :

- when the model is introduced to WES by the USER
- when results are collected from experiments.

From the expert system framework angle, linking the advisor to the controller could be achieved by the development of a model interface, thereby diminishing the user's involvement.

5.2.2 Enhancing the advisor with a model interface

The current consultation model of the advisor adopts the following steps :

(1)- WES is introduced to MODEL by the USER

(2)- WES consults USER

(3)- WES and USER determine problem

(4)- WES advises on experimentation

(5)- WES and USER discuss results

(6)- WES produces REPORT

The introduction of a model interface would reduce the amount of user involvement required and offer an integrated advisory/controller system where steps (1) and (4) could read :

(1)- MODEL introduces itself to WES

(4)- WES carries out experimentation

(i) Introducing the model to WES

Much work has been carried out already in chapter 3 and chapter 4 in order to facilitate this development process. In chapter 4, it was seen how the model is introduced to WES by questioning the user for details about the names of the queues, entities, parameters and default values, seeds and user defined performance measures. A clause "in(learn_name)" directed operations. For example the clause "in(qname)" retrieves the namer of a queue from a user and asserts a fact

qname(Number,[Name],0,0,0).

It is then this fact that is used throughout the consultation. Consequently, if the involvement of the user is to be removed, it must become possible to assert this fact from linking WES to the model.

In chapter 3, in section 3.5 on the "self-explaining" models, a predicate "qnames" was developed to do just that. This predicate worked in conjunction with a FORTRAN subroutine QNAMES that sent these names across, and asserted a fact

qname(Number,[Name]).

All that is needed is to change the assertion to :

asserta(qname(Number,[Name],0,0,0)).

and the objective will be achieved, at the level of obtaining names directly from the MODEL. There is however a small implementation problem at the controller level. The way, the link operates, requires the FORTRAN subroutines to call the PROLOG clauses to be executed and not the other way round. Unfortunately, the consultation is being run from the Prolog engine. A mechanism needs to be devised to cause the Fortran to call Prolog clauses when the time is appropriate.

(ii) Procedural Control of (non-Procedural) Consultation Steps

An approach to the above requirement was to transfer the top level PROLOG inference engine to the FORTRAN side. Intuitively this has pleasing logic. The consultation of the advisor is conducted by the predicate:

```
prolog_engine :-
        meet_model,
          hello_user,
          needs,
          decide,
          controller,
          discus,
          report,!.
```

The sequence of operations is procedurally defined. It therefore makes sense to consider the implementation of their procedural control in a procedural language. The operations which may have a non-procedural execution part would remain executed by a non-procedural language. These operations are the consultation steps and may involve non-procedural reasoning. In terms of implementation, this is achieved by causing the predicate "prolog_engine" to disappear and be replaced by a FORTRAN subroutine ENGINE which would become the top level structure of an integrated PROLOG-FORTRAN program.

```
SUBROUTINE ENGINE
 CALL MEET
 CALL HELLO
 CALL NEEDS
 CALL DECIDE
CALL CONTROLLER
CALL DISCUS
 CALL REPORT
RETURN
END
```

Each subroutine activates its corresponding PROLOG clause. For example :

```
SUBROUTINE NEEDS
 INCLUDE 'LSIM'
 CALL INCOMM                ; initialise link
 CALL CFILL(8)              ; prepare Prolog command
 CALL CSEND                 ; send Prolog command
 CALL CLRC                  ; clear link to receive
 CALL RDI                   ; receive signal from Prolog
RETURN
END
```

Command 8 is supported by :

```
CALL STORE(IMESS,8,'needs')
```

The clause "needs" takes the form :

```
needs :- commend,
         .
         .   all the predicates as before
         .
        writeri(1),!.
```

The idea is for the clause to be executed and then for a flag (here the integer 1) to be sent back to the FORTRAN side to inform the ENGINE that the next step may be tackled.

This implementation, of procedural control of (non-procedural) consultation steps, had one big advantage: it permitted the PROLOG program to remain fairly simple in its execution. Only selected parts of the advisor are used at a given moment. This allowed the adoption of a modular structure. Only the required module needed to be entered into memory in order to execute operations. This was very important at the time of development as the Prolog implementation used was very limiting in terms of program size and its program manipulation capabilities were not without bugs.

Having a procedural consultation engine written in the controller insured that both programs were always in tune. It also illustrates the notion of mixed programming where procedural and non-procedural languages are linked into an integrated program to take full advantages of their respective properties.

(iii) Experimentation with machine reasoning and control of execution

The advisor offers advice on the experiment to carry out, whilst the controller executes particular frames given the appropriate input. Work in chapters 3 and 4 adopted the same expert model and hence used the same experimental frames. In consequence, the advisor holds the input for the controller. Linking advisor and controller at experimentation level extends the work of chapters 3 and 4. Appendix 7-a provides a detailed technical description about how this is achieved.

In the above section (ii), it was seen how the design and execution phase of experimentation is catered for by the subroutines DECIDE and CONTROLLER. The reasoning process is carried out by the DECIDE step and the CONTROLLER routine executes the experimentation. From the intelligent controller point of view, these processes may be seen as the specification and selection of the frame to be executed and can be presented as a modification (see figure 5.1) of the subroutine EXECUTIONER of chapter 3.

(a) Setting up the experiment

The subroutine FRAME1 would be modified to trigger off the clause 'frame_1' presented in chapter 4.

The input values are obtained from the clause "frame_1", which is adapted to send all these values back to the Fortran component.

Having received all required input to the frame, the controller then executes the frame by automatically running the model as before.

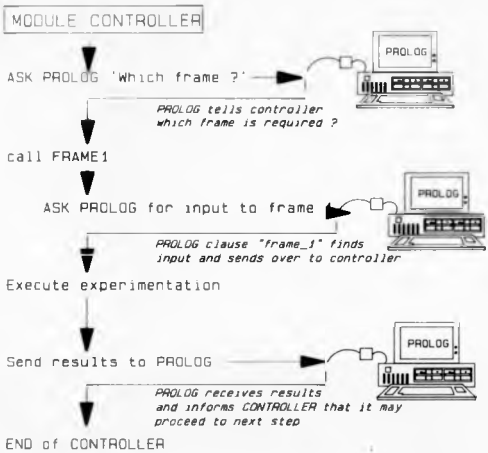


Figure 5.1 : Illustration of the new **CONTROLLER** module

(b) Reading the Results

Thereafter the subroutine **RESULTS** controls the return of all the results to the **PROLOG** advisor. Again, a clause "results" is associated with the subroutine **RESULT**, which for example, will cause the clause 'resbqu' to stand ready to receive the utilisation of the specified queue.

The other results are read back in a similar fashion using similar subroutines and clauses, and the analysis is carried out as before by the advisory side of the system. A question that needed to be resolved was whether the advisor or the controller should process results to obtain means and standard deviations. Initially, the low level intelligent enhancement dictated that only aggregate results need be stored. But experience with the advisor revealed that to make use of the perfect memory concept, individual results need to be stored. In this respect, it was decided that the advisory component should perform the analysing. In retrospect, the primitive numerical

manipulations of Prolog have made this a dubious choice as processing is very slow.

(iii) Assessment

The outcome of this linking operation is to further remove the user from the experimentation process. This integrated system carries out the operations of intelligent reasoning and control of execution. This combines the support provided by both the controller and the advisor. The user essentially becomes a spectator as the system performs tasks of strategic reasoning and model execution.

5.2.3 Problems and Limitations of Parallel Linking

The parallel linking of the advisor to the controller served the purpose of illustrating how the user can be supported in both the execution of his model and even the design phase of experiments. This exemplified the notion of remote execution, with the model sitting in one machine and the user interface with the advisory system in the other machine. This environment however has some practical shortcomings in real application implementation. There are not always two dedicated micro-computers available for use by one person only. It would be better if the model, controller and advisor could all be kept to one machine.

A second limitation with the parallel link of the advisor and controller is the operation of the top level consultation engine. At present the procedural aspect of the consultation is directed by the FORTRAN superstructure activating the appropriate functions when required. Although this set up was found desirable for the Prolog implementation used, it is in fact limiting for a system whose behaviour may not be procedurally defined in the light of increasing knowledge and added behaviour patterns. Indeed, the procedural structure to the consultation need not always hold as new behaviour patterns may be added. This was addressed at the end of chapter 3, and in some sense the notion of procedural control of (non-procedural) consultation steps may appear contradictory to the concepts presented in chapter 3 concerning non-procedural control of execu-

tion. Though these extra behaviour patterns could be implemented with the present link (with some degree of complexity), there does not appear to be a strong case for a non-procedural path to be catered for by a procedural language. It is felt that the consultation is best controlled from the PROLOG side.

A third consideration is that the current implementation of the link betrays the original ideas portrayed by Flitman for parallel execution. Indeed, experimentation using the link is in effect following a sequential cycle of operations : the advisor decides and then the controller acts before the advisor decides again. Only one program is active at a given time while the other is suspended. There is therefore no run-time justification for two machines.

For these reasons, work was carried out to generalise linking procedures that could be held sequentially on a single machine.

5.3 Sequential Linking of Advisor to Controller

5.3.1 Objective

The objective is to condense the integrated system presented in section 5.2. which exists in an environment of two micro computers that communicate using a RS232 serial link. One machine holds the controller and model, the other holds the advisor and user interface. The goal is to halve the number of machines so that the programs relating to the controller and to the advisor share the same machine. A new link is required and it is anticipated that sequential linking be used. This would mean that one program is to be called, loaded into memory, executed and removed from memory before the process may be repeated with the other program. In section 5.2, the linking procedure of advisor to controller was seen as carrying out tasks of specification input (from the advisor to controller) and then of results output (from controller to advisor). The centre of attention in the development of a sequential link will lie in the data transfer of input and output from one program to another. Careful consideration is also to be given in finding an appropriate cycle mechanism to direct calls of one program to another and automate their execution, so that the user does not have to be involved in this process of program manipulation. The user should not need to do more than he did with the parallel link.

In section 5.3.2, the operations for combining the advisor and the controller will be described. In section 5.3.3, the actual input and output data transfer issues are resolved with a discussion in section 5.3.4 of the implication of a sequential link to the controller, advisor and simulation model structure.

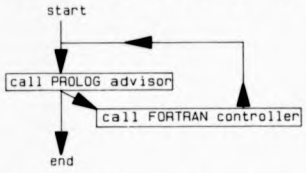
5.3.2 Sequential Operations

The parallel link described in 5.2 was achieved at the assembler level of computer coding. The sequential link is devised at the higher level of operating system commands. The operating system was MSDOS and the link was achieved through batch file programming. This batch file

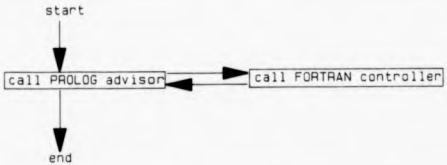
controls the operations of calling the appropriate programs at the right time.

(i) Observation of Sequential Operations

A pattern can be detected in how the programs are called as illustrated by the flow diagram of operations below.



The first operation would be to call the advisor into memory for a consultation. Eventually, the advisor may either signal the end of the consultation or request some experimentation to be carried out. In the latter case, this implies that the CONTROLLER must be put into memory. (How the controller and advisor communicate input and output is left to the following section.) At the end of experimentation, execution will need to return to the Prolog advisor, and this is where difficultises arise. If continuity is to be kept in the consultation, execution should re-enter at the point of departure. Fortunately, this point will be the same every time experimentation is undertaken, as the first priority of the advisor will be to collect results. Therefore a better flow diagram would be as follows.



The above flow diagram suggests two different entry points to the PROLOG program. This in turn implies that a mechanism must be found to differentiate between the two possible alterna-

tives. This is achieved by making use of the initialisation facilities offered by PROLOG. Such initialisation facilities, as presented here, are available in most PROLOG implementations. (ARITY-PROLOG was used for the integrated expert).

(ii) Implementation tools for Sequential Linking

Initialisation may be achieved by automatic execution of a file called PROLOG.INI. The purpose of an initialisation file is to automatically load the appropriate program and execute any number of declared predicates. Therefore two initialisation files could be created to cater for :

- initialisation of consultation,
- re-entry after experimentation.

The first case would load the Prolog advisor and activate the top level predicate to start the consultation process. The second case would reload the Prolog component and activate the next clause to be executed. After experimentation, there will always be activation of the clause "results" to read the results obtained from experimentation. Since, the consultation process follows sequential steps in principle, it easy to reactivate the consultation process.

A feature offered by most Prolog implementations is the capability to save the current status of a program during its execution. This means that the blackboard application component can be saved as it is. In ARITY-Prolog, the command "save(X)" performs this task. This status of the blackboard may be brought back into memory by a predicate "restore(X)". This permits resuming the consultation with all asserted facts of knowledge saved.

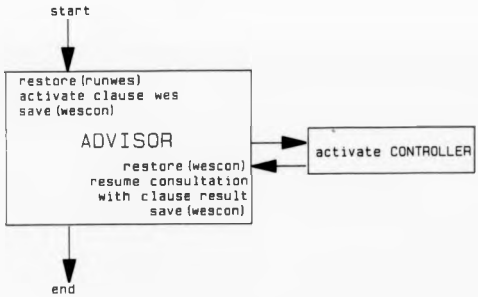
This "save(X)" facility is also useful to save on compilation or interpretation time. The implementation of the link proposed considered the creation of two status files called "runwes" and "wescon". The first stores the PROLOG component of the integrated system ready for use. The top level command for this file is "wes" which activated the consultation predicate. The second status file ("wescon") is created during consultation and immediately prior to experimentation when the Prolog component is left. Reentry is through a clause "results".

Making use of these implementation tools it is possible to write two initialisation files called "startwes" and "morewes".

```
startwes :
        :- restore(runwes).
        :- wes.


morewes :
        :- restore(wescon).
        :- results.
```

The complete flow diagram of sequential operations may be then formalised as illustrated below.



(iii) Operation Control through Batch File Programming

The user's involvement is no longer needed to boot the advisor or controller up. The user keys into the whole integrated system by activating the batch-file program. This is achieved through the command "model". This command "model" takes one parameter : the name of the model that is to be experimented with. Once this command has been activated, the system takes over and the user becomes the patient. The batch file program may be found in appendix 7-b.

5.3.3 Input and Output communications from the Advisor to Controller

The previous section (5.3.2) has illustrated the feasibility to have a control process at the operating system level that would support the activation of the Prolog and Fortran components at the appropriate time. What still needs to be established is how the advisor can input to and receive output from the controller.

The parallel link stored common data structures at assembler level. The sequential link is to share ASCII data files at program level. Although a single file will suffice, it was decided to use two files for communication purposes : one to be the input file to the controller, one to be the output file from the controller. This decision was taken for development purposes. It was easier to check input and output at the same time. A single file causes either input or output to be overwritten by subsequent output or input.

(i) Input from the Advisor to Controller

In chapter 3, it was seen how an experimental frame could be encoded so as to be executable given some form of input. In section 5.2, it was described how input could be obtained from the advisor to operate the controller. With some adjustments, it will be seen how input from the advisor can be transferred using this new linking procedure.

The file "HI" is used as the transfer file from Advisor to Controller. It carries data from the clause "frame_1" to the subroutine "FRAME1". By carefully matching up these two clause and subroutine, input may therefore be read sequentially. Rather than receiving one input at a time, the controller receives all input at once. Required input are:

1. Flag (values 0 or 1) to indicate type of experiment
2. Flag (values 1,2,4) to indicate mode of runs (batch,normal)
3. Range acceptance for runs (in time units)
4. Number of subroutines in the model
5. Run in time to be used

6. Length of runs to be used
7. Number of parameters to consider
8. Number of queues to consider
9. Number of performance measures to consider
10. Number of seeds to consider
11. Values of the parameters to use in experiment
12. Values of the seeds to use in the experiment

This input will permit the controller to run the experiment. Appendix 7-c provides more details on this communications protocol.

(ii) Output from Controller to Advisor

This output is transferred using file "BYE". The results of the experiment are carefully stored sequentially so as to ensure that the advisor can pick up the appropriate results in turn about :

- the average utilisation of all queues,
- the average time-in-the-system of entities
- the value of the performance measures.

The clause "results" from the advisor retrieves this data from the file "BYE".

5.3.4 Implications of the Sequential Link

The switch from the original parallel link environment has necessarily had some implications on the various components of the integrated system. These are the advisor, the controller and the model. The implications are reviewed in turn.

(I) The Advisor

The advisor requires changes in two parts : clauses "frame_1" and "results". "frame_1" now write to a file :

```
frame_1 :- write('WES is busy running the following experiment with
           your model'),
              .
              .
              .
           open(H,'hi',w),          ; open file to write input
              .
              .
              .
           write(H,X),              ; write input
           haltwes,!.
```

The clause "haltwes" is important as it will save the present status of the consultation. As explained, this means that when execution returns to the Prolog side, WES will know the name of the queues and other model components, the nature of the problem under investigation, the experimentation undertaken and the results already collected.

```
haltwes :- save(wescon),
           halt,!.
```

At reentry time, the clause "results" is called first :

```
results :- open(H,'bye',r),          ; open file to read output
           action(1,E,[rutim,llastrun,...)
              ...
           ps(resbqu),               ; read queue utilisations
           ps(resbpv),               ; read performance measures
              .
              .
              .
           next,!.                   ; resume consultation
```

For example, queue utilisations are retrieved:

```
ps(resbqu) :- qnamenu(Y),                    ; number of queues
              count(A),qname(A,[X],Za,Zb,Zc),  ; consider next queue
              read(H,Qu),                    ; retrieve utilisation
              assert(wfac(900,E,[X],Z,Q,0)),   ; save as a fact
              Ass is A+1,Ass == Y,!.         ; check if last queue
```

All results are asserted in the same way as previously. Resuming the consultation processes is achieved by the clause next.

```
next :- controller,
        discus,
        report,!.
```

The experimentation process is controlled from these clauses. The inference engine operations are continued from these clauses.

(ii) The controller

The controller was in fact rewritten because most of the code used in the previous version was impregnated with command calls that were specific to the previous link. The superstructure on the Fortran side now comprises a number of subroutines that perform specific tasks.

SUBROUTINE SIMWES : reads the values from the file 'hi' (input from advisor).

SUBROUTINE RINEXP : determines stopping conditions.

SUBROUTINE WESSUB : determines starting and stopping of monitoring operations.

SUBROUTINE RISTOP : monitors run-in time investigation experiments.

SUBROUTINE WESTOP : ends the experiment and collects the results that are output to file "bye".

SUBROUTINE UTILIS : computes the utilisation of each queue after each time advance.

SUBROUTINE QUEWES : computes average utilisation at the end of experiment.

SUBROUTINE WESMON : starts the monitoring of performance measures.

SUBROUTINE TBCOND : receives a list of starting and stopping conditions for transient behaviour investigations.

SUBROUTINE TBGO : starts the monitoring in transient behaviour investigations.

SUBROUTINE TBSTOP : ends the monitoring in transient behaviour investigations.

Listing of the sequential link CONTROLLER may be found in the WES technical documentation.

(iii) The model

A number of small changes are necessary in order to use a simulation model with WES. These are presented in appendix 7-d. Two small inclusions are needed to the main program structure to insure initialisation and time advance operations for the controller processes. More significantly, the issue of passing variable values between CONTROLLER and the simulation model is addressed differently to the parallel link where equivalence statements were used. In the sequential link, the user must include subroutines in the standard form describe below to transfer values from the general CONTROLLER variables (IWES(X), SWES(Y), IPWES(Z), IQWES(W)) to the model defined variables.

1. Parameters. The user must decide which are the parameters of the model. They must be set to the value IWES(JJ) which is a controller variable by including the following subroutine.

```
      SUBROUTINE PARWES
       INCLUDE 'WESSIM'
       INCLUDE 'common_file'

Either  CALL SETATT(ARATES,1,IWES(1))
        CALL SETATT(ARATES,2,IWES(2))

Or      PARAMETER_3 = IWES(3)
        PARAMETER_4 = IWES(4)

      RETURN
      END
```

2. Seeds: it is necessary to ensure that randomly generated numbers have a variable seed. This is required to replicate experiments. The variable seed should be set to the value of SWES(JJ). The following subroutine must be added to user's program :

```
SUBROUTINE SEEWES
 INCLUDE 'WESSIM'
 INCLUDE 'common_file'
 ISEED_1 = SWES(1)
 ISEED_2 = SWES(2)
RETURN
END
```

3. Queue utilisation : the size of the queues that are to be considered must be set equal to IQWES(JJ).

```
SUBROUTINE QUTWES
 INCLUDE 'WESSIM'
 INCLUDE 'common_file'
 IQWES(1) = ISIZE(QUEUE_1)
 IQWES(2) = ISIZE(QUEUE_2)
RETURN
END
```

4. Entity time-in-the-system variables and performance measures are considered in a similar fashion as for example in the subroutine RESWES below for performance measures

```
SUBROUTINE RESWES
 INCLUDE 'WESSIM'
 INCLUDE 'common_file'
 IPWES(1) = PERFORMANCE_1
 IPWES(2) = PERFORMANCE_2
RETURN
END
```

From the user's point of view only a few changes are required. Theses changes were found easier than the equivalence statements of the parallel link.

5.3.5 Limitations of Sequential Link

The main limitation of the sequential link over the parallel link is the fact that only data can be transferred. With the parallel link, it was possible to remotely execute clauses from the FORTRAN side. This was used previously, when the consultation was conducted from the FORTRAN ENGINE. Since it was eventually decided more suitable to control the consultation entirely from the Prolog side, this limitation has not proved troublesome at all.

In section 5.2, the type of input to the advisor from the model was divided into two categories :

- input during model introduction

- input during experimentation

The latter type has been well catered for by the sequential link. The former however has been ignored. There is no longer any automatic learning of names by WES. Though this could still be done, it was felt that it is important for the user to tell WES himself about the characteristics of his model. This ensures a basic common understanding by both user and WES of the features of the model that they are about to deal with.

5.4 Enhancing Machine Expertise

5.4.1 Objective

The work presented so far in this chapter has covered the foundations for an integrated system that can assist in the experimental design and analysis phase of a simulation study. This integrated system supports a consultation framework whereby users can address the system with particular queries about their model and let the system carry out the relevant experimentation and analysis operations.

Rules to operate the "solution seeking" phase have been input to carry out the tasks implemented. These rules are fairly simple and though they work satisfactorily within their own limits, they might not be considered as real expertise. As defined in chapter 4, they are only default rules. The system developed will not claim to be an expert. However, it is hoped that in the course of this section 5.4, suitable mechanisms may be developed to allow the acquisition and implementation of knowledge that tends towards expertise. The objective is to show that the integrated system developed does support the structure for an "expert" experimenter if time was spent to obtain individual expertise from a (large) number of human experts.

In section 5.4.2, the interest will lie in the differentiation of model-type dependent and model-type independent rules. Section 5.4.3, will consider a mechanism to allow users to input their own rules. And section 5.4.4 will explain how other frames could be implemented.

5.4.2 Model-Type Dependent and Model-Type Independent Rules

In this section, a look is taken at the idea of having rules that are specific to a certain type of model. For example, a class of models may be simulation systems that refer to manufacturing applications (factories, workshops). Another class may refer to service industries (shops, banks). For each class, a number of better rules may be found to the default rules of WES. These default rules are considered model-type independent. They refer to rules present in the rule-base

(pred_rule_(X), eval_rule_(X), comp_rule_(X), sens_rule_(X), tran_rule_(X), f_rel_rule_(X), optim_rule_(X), exp_design_rule_(X), run_in_rule_(X), run_time_rule_(X), num_run_rule_(X)).

Model-type dependent rules may be stored as a set of extra rules to be used when a model of that type is being used. Prolog makes it very easy to introduce a new set of rules into the rule-base. These rules only need to be interpreted before the main program. An example probably best illustrates the point being made.

(i) Example

Consider the die-shop model. The die-shop may belong to a class of models called "manufacturing models". A particular feature of manufacturing models may be that a week's worth of simulated run-in time is sufficient to obtain a "good enough" truncation point. This fact may be embodied as a rule (say run_in_rule_(manufacturing(X)).

When the die-shop model is loaded for experimentation, a facility should exist for manufacturing rules to be loaded too. Then when experimentation occurs, and a run in time is sought after, this specific run_in_rule(manufacturing(X)) will be called up in preference to the model-type independent default rules of the system.

Expertise may thus be introduced by building up a number of mini-rule-bases that are specific to the types of models.

5.4.3 Mechanism to input user's own expertise

In the simulation package used (MICROVISION) for the model presented in this system, there is an "OWNINT" device (OWN INTERACTION) that allows a user to carry out his own interactions. He can make his own changes to the model in a "controlled" manner: ie. only acceptable changes to the model may be investigated. The objective of the concept presented here is to offer the expert/user an "OWNRUL" device that should allow him to experiment with his own knowledge. An experienced user is offered the facility to encode his own "expertise" to be used by the system. In the same way as model-type dependent rules can be used with the

system, expert's own rules can be loaded in to complement or replace the system's rules.

There is some difficulty in preparing a mechanism that will intake any type of rule from a user, expecting the system to understand and incorporate it at once. The "default" rules are all encoded in Prolog in a form that permits direct understanding by the inference engine. It is also not practical to expect a user to know which piece of code will represent what he is trying to say. For this reason, two alternatives are offered to a user for inputting his own expertise :

- inputting rules from antecedents,
- inputting free-format rules.

The first approach may be limiting in the nature or form of the rules but is guaranteed to be understood by the system immediately. The second method is free from any format constraint on the rule formulation but may require user involvement when fired. Implementation of the concepts presented form a supplementary sub-module to the advisor and may be found in the WES technical documentation under sub-module c.

(I) Rules from Antecedents

All rules considered must be expressible in the form :

IF conditions are true,
THEN conclusions apply.

The idea of inputting rules by "example" is to present a number of possible conditions and conclusions to the user. These conditions and conclusions have antecedents in existing rules. The user is asked to select the conditions that he wishes to use for his rule, and then similarly the appropriate conclusions.

The possible conditions and conclusions are grouped by rule type. If the user selects to input his rules from antecedents, he is asked to indicate the category of the rule by choosing from a list that reads :

1. Prediction rule
2. Experimental design rule
3. Evaluation rule
4. Comparison rule
5. Sensitivity analysis rule
6. Optimisation rule
7. Functional relation rule
8. Transient behaviour rule
9. Steady state rule
10. Run time rule
11. Number of runs rule
12. Other

According to his choice, the user is presented with a list of conditions that are used in other rules of that type. All these conditions are stored in facts of the form :

```
qfac(Number,['English meaning'],['Prolog code'])
```

The database of such facts is partitioned according to rule type. The user is presented with the English translation. If a condition is chosen, then the Prolog code for that condition is stored. When all conditions and conclusions have been selected, the rule is presented to the user for approval and saved, written in Prolog code, ready for immediate use.

Example:

Suppose a run time rule is to be selected. A list of conditions might be :

```
qfac(227,['run-in time less than 1000'],['run_in_time_is(Xi),Xi<1000']).
qfac(228,['this is an optimisation problem'],['problem([o])']).
qfac(229,['the model has less than 3 type of entities'],
                        ['enamenu(Xe), Xe<3']).
```

Conclusions may be :

```
qfac(727,['run time is 500'],['assert(run_length_is(500))']).
qfac(728,['run time is 5000'],['assert(run_length_is(5000))']).
qfac(729,['run time is twice run in time'],['run_in_time_is(Xi),
                B is 2*I, assert(run_length_is(B))']).
```

The user may have chosen condition 1 and 2 and conclusion 1. Before the rule is input to the rule-base, it will be presented :

Rule is own(run-time-rule)

IF
- this is an optimisation problem
- run-in time is less than 1000

THEN
- run time is 500

If the rule is accepted then the code is saved in the form below in a file that holds the expertise of this particular user.

```
own(run-time-rule) :-
    problem([o]),
    run_in_time_is(Xi),Xi<1000,
    assert(run_length_is(500)),!.
```

The rule is ready to work as WES can pick up all these values.

(ii) Inputting Free-Format Rules

The user is asked to classify the type of the rule he is inputting. He will also be presented with the list of possible conditions and conclusions that he may (but not necessarily) use that are associated with the rule type. He has to type in each condition and conclusion, which WES scans in turn. If it corresponds to an existing condition or conclusion, the Prolog code is recorded and will be used in the rule. For cases when there is no antecedent in WES, a user interaction point must be implemented. This means that the user must be made to answer a particular query or activate the appropriate conclusion. For example, suppose in the example above, the user inputs the following modification :

IF
- this is an optimisation problem
- run-in time is less than 1000
- average time-in-system of entities is less than 50

THEN
- run time is 500

This will be encoded as :

```
own(run-time-rule) :-
      problem([o]),
      run_in_time_is(Xi),Xi<1000,

      write('Is it true that ')
      write('average time-in-system of entities is less than 50'),
      read(E),
      (E==y;E==yes),

      assert(run_length_is(500)),!.
```

When a rule input is executed, the control will have to obtain from the user an answer to the question about the average time-in-system of entities. It is not very hard to find appropriate code to retrieve this information but it requires a system analyst to do so. This existing facility permits observation of user/expert's reasoning process and permits refinement of each rule at a conceptual level before it is formalised and implemented in Prolog code. Section 7.4 presents a set of rules that were initially prepared using this facility in order to investigate the use of uncertainty factors.

(iii) Meta level control of new rules

Two issues arrise from the addition of new rules :

- completeness of experimental input
- execution timing

(a) Completeness

The issue of completeness of experimental input stems from the problem that when users develop their own rules, they may not always include all the necessary specification components for an experiment. A rule is therefore needed to check for completeness and retrieve the missing specification items.

Simulation experiments are run from prediction rules. For a pred_rule_(X) to apply, it must first have been established :

- what is being monitored

- run in time

- running time

- number of runs (first run, last run)

- assertion of experiment to carry out

So if a own(pred_rule) has been used then a check rule might be :

```
pred_rule_(0) :-
    own(pred_(1)),
                    ; check to see if own prediction rules
                    ; have been used

    (run_in_time_is(lrutim);run_in_rule_(X)),
                    ; if a run-in time have not be
                    ; identified then find one using
                    ; run_in_rule_(X)
    (run_length_is(lhatim);run_time_rule_(Y)),
      (exrun(lrun);run_num_rule_(Z)),
                    ; same for run length and number of
                    ; runs
      run_in_time_is(A),
      run_length_is_(B),          ; set up experiment
      exrun(C),
      expnumber(Hh), Zz is Hh - 1,
      (quexp(_,_),qoexp(_,_),pvexp(_,_)),
      assert(action(1,Z,A,B,E,F,lmqu,lmqo,lmpv)),!.
```

This rule checks for the existence of specification items, and if they are not present, appropriate rules are fired to derive them.

(b) Execution timing

The issue of execution timing relates to the problem of knowing when to fire these rules. Meta-level knowledge is already implemented to guide the system according to problem under investigation. Meta-level control is required for every new rule that is implemented. WES needs to know if this new own(rule) should be called :

1 : before all rules,

2 : before WES rules for problem under investigation

3 : after WES rules for problem under investigation

4 : after all rules

A predicate ow(call_rule) is activated when a rule has been implemented to discover this. The choice is as above and is stored in facts of the form :

```
orcall([Name_of_rule],E)
```

where - Name_of_rule is names of rules
- E is 1,2,3 or 4 according to choice

Meta level control rules can be inserted in the form :

```
meta-rule :- orcall([X],1),
            ownrule([X]),                ; before all rules
            fail,!.

meta-rule :- problem([X]),
            (X==p,pred_rule),
                .
                .
                .

meta-rule :- orcall([X],4),
            ownrule([X]),!.              ; after all rules
```

Within each class of rules, two new rules can be inserted . eg.

```
pred_rule_(X) :- orcall(['pred_rule'],2),      ; before prediction
                 ownrule(['pred_rule']),!.     ; rules


pred_rule_(X) :- orcall(['pred_rule'],3),      ; after prediction
                 ownrule(['pred_rule']),!.     ; rules
```

This ensures that ownrules are looked at, at the appropriate moment. Further refinement may be required if rules should be used between certain of WES default rules.

(iv) Loading file of new rules

The initial consultation phase of WES is modified to allow a user to specify if a separate rule-base should be used in conjunction with the model selected. See appendix 13-a to see how this is implemented.

5.4.4 Additional Frames

During the early development of the advisor and integrated system, only one experimental frame was implemented. Expertise of the system could be increased at the controller level by adding new frames. Furthermore, it has been noted that only heuristics have been used in this work where probably more suitable algorithms would constitute better expertise.

A certain number of new frames were added in the integrated system to help achieve tasks of finding appropriate run in times. In particular, a monitoring device was implemented in the controller to determine when the aggregate utilisation of the system stabilised in order to establish appropriate run in times. The addition of these extra control structures is discussed in chapter 7, where it will be seen that the proposed system accepts changes to include:

- a new simulation environment;
- new expert behaviour (ie. new experimental frames);
- new quantification variables.

5.4.5 Usage Experience

The mechanisms presented for rule development were tested by the author who sampled informally a more expert "simulationist" to develop a sample rule-base. Although the mechanisms appear robust, the author, being the system analyst, rarely made subsequent use of this facility to refine existing rules since he was always able to write direct code.

5.5 Conclusion

In this chapter, an integrated system was presented that linked the controller of chapter 3 to the advisor of chapter 4. First a parallel link was used that offered a procedural control ENGINE to non-procedural consultation steps. Next a sequential link was used which performed the control process from the Prolog side. This achieved a system that reduced the level of user involvement, during the task of experimentation, to simply specifying the nature of the problem to consider. Experimental design and analysis is carried out by the system and the execution of the experiments is automated. The framework appeared fairly robust as implemented rules, though simple, achieved required objectives. Although, it was not the objective of the system to achieve expertise, it was nevertheless felt necessary to develop facilities that would support the acquisition and implementation of expertise from experienced users. Notions of developing rules from antecedents or with free-format were implemented.

The outcome has been the development of a framework of an environment that has some machine intelligence and machine control in order to support the practice of simulation experimental design and analysis. Appendix 8 provides a summary of the system's structure in diagram form.

This chapter has demonstrated the potential of an expert advisor using procedural and non-procedural programming. It would be expected that multi-processor systems will be able to exploit this parallelism.

End of Chapter

Chapter 6

A Teacher - Enhancing Understanding using Machine Knowledge

6.1 Introduction

In the previous chapter, facilities were developed to permit experts or experienced users to input their own knowledge into the system's rule base. The objective was to support a framework that could help turn the integrated system into an "expert" in its own right in the field of simulation experimental design and analysis. Augmenting the system's knowledge into expertise can only increase the level of support that may be provided to a user. Paradoxically, if the user could acquire and understand this knowledge, his future needs for support may be reduced. In this chapter, the roles are thus reversed : the system is no longer the pupil of the expert user, it becomes a teacher for the occasional user. The objective is now to enhance human understanding of the experimental design and analysis process through machine knowledge. Two complementary approaches to teaching are considered : explanation and demonstration. It is not argued that these teaching mechanisms provide superior pedagogical results over conventional teaching methods. The issue only surrounds the extraction of this concentration of system's knowledge for the benefit of the user.

Traditionally explanation facilities are implemented in expert systems for:

- debugging the knowledge base,
- validating rules (Davis, 1977);
- understanding the system's actions,
- justifying the reasoning of the system, (Shortliffe, 1975);
- helping to educate the user.

In section 6.2, much work is spent on implementing a facility that addresses these objectives, focussing on the educational goal. Conventional approaches to explanation facilities are found to

be unsatisfactory and a new methodology is proposed for integrated systems that not only advise but control the execution of external devices (eg simulation models).

Human understanding can be increased through demonstration (Stephenson,1982). A facility is implemented that conducts tutorials of investigations to provide insight into what is happening and why. In chapter 7, some experimental results are discussed about the use of this facility. In this chapter, the term "user" refers to the author's colleagues who kindly accepted to test the system during its development stage.

6.2 Teaching through Explanation

6.2.1 The MYCIN mould

The characterisation of "MYCIN mould" reflects the influence of the MYCIN explanation facility which is probably one of the best treated in the literature. R.Davis (1977) published much of his PhD work on its development. It is not the objective of this section to review his work but only to present some of the issues and principles that inspired many other explanation facilities.

(i) Who is explanation for ?

There is a need to make the explanation fit the recipient as there is a need to vary the depth of detail dependent upon the knowledge that the recipient has.

(ii) How is it achieved ?

At the simplest level, explanation, in a production rule base system, can consist of presenting the basic rule or free translation of the rule. But for a system to provide good explanation, the system needs to provide an explanation of :

- how it made a certain decision,
- how it used a fact/piece of information,
- how it will try and deduce a fact,
- why it did not use a fact/piece of information
- why it failed to make a certain decision,
- what its present state of knowledge leads it to consider as the most likely hypothesis.

(iii) Requirements for a good explanation facility

- maintaining a history mechanism to provide details of how reasoning steps were carried out;

- access facilities to the dynamic knowledge base;
- access facilities to the static rule base;
- a mechanism to extract procedural information from meta-level rules;
- a model of the user in order to provide explanation at an appropriate level of complexity;
- a multi-granular explanation tag on the rules;
- a context sensitive translation mechanism.

(iv) Problems

Such systems if implemented are quite complex. Unfortunately, there are some known problems with even sophisticated implementation of the above :

- Lack of adequate "support knowledge" giving an explanation of the mechanism or associational links that explains why the conclusion section of a rule follows from its premise.
- Ability to adjust the explanation to the context in which the explanation is requested : the same question asked at different points could require different explanations.

These weaknesses, it would appear, stem from the assumptions behind such explanation facilities. Davis (1977) makes two assumptions. Firstly, *"a recap of program actions can be effective explanation as long as the correct level of detail is chosen"* and secondly *"there exists some framework for viewing the program's actions that will allow them to be comprehensible to the observer"*. These assumptions possibly derive from an emphasis to address the first four objectives of traditional explanation facilities, as outlined above, in preference to educational issues that may result as a benefit rather than as an objective.

These problems, of support knowledge and contextual requirements, were carefully considered during the development of the explanation facility that supports WES, whilst placing pedagogical performance first. The development approach was to start from existing methodologies for explanation facilities and examine their appropriateness to WES. Refinement or change would result from empirical evidence.

6.2.2 "Single Explanation" Facility

(i) General considerations

The structure of WES hinted that different steps in the consultation framework may require different types of explanation. The different steps were :

1. - initialisation of consultation
2. - problem investigation
3. - problem solving
4. - problem solution execution
5. - discussion of the results
6. - report generation

Prior to the problem solving activities which execute the rule base, it appears that issues surrounding support knowledge and context are best addressed through explanation of the concepts used in the early problem investigation phase. However once rules have been used, there may be value in presenting them to the user.

(ii) Interactive requests for explanation

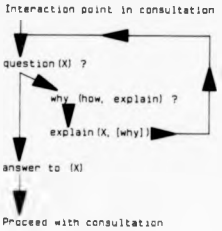
The initial work was simply to make the system aware that explanation may be required. It had to be made possible for the user to request some form of explanation and the system to realise this. A general approach was taken about when and how the user could request information :

- when : at every question point;
- how : by asking :
  - why : the system requires this information
  - how : the user can answer this question
  - explain : the situational context.

This principle is common to most explanation facilities, although the use of the HOW facility to assist the user in using WES is unorthodox. The first requirement is to put a check on every input from the user to enable the system to detect whether this keyboard input constitutes the answer or is a distress call for help. This was implemented using a generalised "question(X)" clause.

```
question(X) :- quest(X),
               read(Answer), asserta(query(X,[Answer])),
               helpcheck, hc1,!.
```

Every interaction point in the consultation is signaled by a question(X) call. For every question, there is a corresponding quest(X) stored in a question database (see submodule 1 in the WES technical documentation manual) holding the actual English phrasing. The answer is read and stored before a clause "helpcheck" performs operations to check the answer. This early implementation had a very simple pattern :



This translates into :

IF : answer was "how", "why" or "explain"

THEN : provide explanation by activating a clause of the type :
explain(X,['how']),
or explain(X,['why']),
or explain(X,['explain']), depending on the request;

ELSE : proceed with consultation.

(iii) Contextual explanations

These clauses (explain(X,[_])) contain textual explanations to the question asked. Since every question is associated uniquely to a given point in the consultation, the explanation may be provided with contextual understanding. If concepts are discussed, they will be referenced to the part of the consultation where the question originated from. For example :

```
explain(1,['how']) :-
  write(' This explanation tells you how to answer question 1'),!.
```

In implementing this approach, a question database was set up with an extensive explanation database. This approach accomplished elementary explanation. It ensured that at any question/interaction point, the user was able to request some help relating to either :

- why the system required this information, or
- how he could provide this information, or
- explanation of the contextual concepts.

(iv) Limitations

Although this facility provides context to explanations which are geared towards support knowledge, it is limited as it assumes that the user will understand and be satisfied with a single explanation: more a utopic hope than a reasonable assumption. The next stage was to permit further queries into a problem area.

6.2.3 Multiple Queries

The "multiple queries" concept is an extension to the "single explanation" facility that permits queries about queries. The user is asked if he is satisfied with the explanation provided. A negative answer causes a prompt for him to choose between "explain", "how" or "why". These options refer to the refinement required. This idea is paralleled with the MYCIN view of an explanation facility as a tree which can be ascended using "how" or descended using "why".

The MYCIN approach goes from node to node according to past history of the consultation pattern which is strictly production-rule driven. It was felt that this approach would not be suitable in the initial consultation conversation of WES, as the educational value of the systems behaviour appeared secondary to the concepts presented in the question themselves. The alternative investigated was to direct WES to the next source of explanation. This meant the creation of a select_next_query database to support the original explanation facility. The facts of this database have form :

select_next_query :- (X,[A],[B],Y)

where :

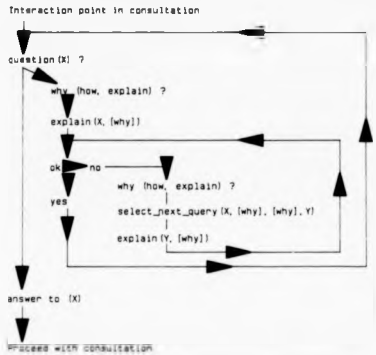
X : is number of last explanation,

A : is nature of last query (how, why, explain),

B : is nature of new query (how, why, explain),

Y : is number for next explanation.

When the select_query database is consulted, three of these parameters are known : X, A and B. When a select_query fact has been found, a new query is presented to the explanation facility : ie query(Y,[B]), and the appropriate clause explanation(Y,[B]) is fired. Operations flow as follows.



The outcome of this implementation is an explanation facility that permits the user to ask the system to refine its explanation by going up and down the explanation tree. Each further explanation selected using the select_next_query database will relate in context to the previous explanation, therefore preserving contextual logic. To make this system work adequately, a couple of points need to be considered :

- what happens at the top and bottom of the tree ?
- how much refinement of an explanation is possible or desirable ?

(i) Buffer explanations

The top and bottom of the explanation trees will be reached if the user requests continuously "why" or "how". For example if after a multiple "how" sequence, the user ends up with :

"Just type 1,2 or 3 to select your choice."

and he request a further "how", a buffer explanation is required. Buffer explanations have form :

"No more explanations available..."

A further request from a buffer explanation will only cause the buffer explanation to be repeated.

(ii) Curbing exponential growth

The second point, relating to how much refinement is possible or desirable, considers the growth of such a facility. Consider figure 6.1.

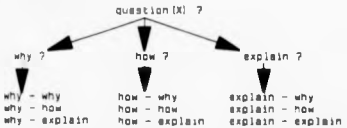


Figure 6.1 : Exponential growth of an explanation facility

From every starting question, there is a need of nine facts to get to second level explanations. In fact, in theory, every further level of query increases the number of explanations by a factor of 3.

The growth is a cubic exponential. In practice, to prevent an explosion of the explanation database, it is possible to shortcircuit the growth by converging onto the same refinement explanations from different starting question from the consultation (see figure 6.2).

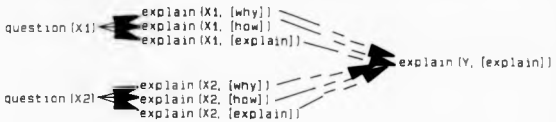


Figure 6.2 : Converging explanations

This implies eventually a move from particular situational and contextual explanations to general support knowledge explanations. This means that although the first explanation provided will be contextually right and therefore will be so expressed, subsequent queries must take a general tone if they are indeed to be general. In practice, this has amounted to the implementation of a large number of explanation clauses that can only be addressed during further explanation requests. This may be viewed as a general explanation tree that may only be entered after contextual information has been removed from the explanation (see figure 6.3).

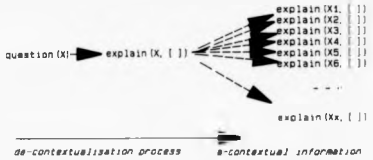


Figure 6.3 : de-contextualisation process of an explanation

(iii) Limitations

"Multiple Queries" attempted to address both issues of contextual explanation and support knowledge. However, there are a number of serious limitations to this system.

- Each question requires a considerable amount of code that is specific to each question : ie select_query database.

- If support knowledge is required, it may not be possible for the user to obtain it. He has no means of telling the system what he wants. And the system only has a limited set of refinement steps. Furthermore, at each step, the potential for further explanations is increased.

- Once you have gone up the tree using "how" (or "why"), you can rarely return to a given explanation.

- There is still no teaching about the system's reasoning: the rules are not explained.

At this stage of the work, the most serious of these limitations appeared the fact that it was difficult to get the system to converge on the user's real problem. Since the explanation facility was not dynamic, it was considered to replace it with an information system.

6.2.4 An Information System

The objective of the information system was to overcome the shortcomings of the multiple queries framework by formalising multiple investigation routes at any explanation point. The previous queries (how, why, explain) were replaced with a single "help" request. Such a call invoked the information system. The system has a tree structure (see figure 6.4), where control moves from node to node, up and down the tree. At ground level, text files provide explanations on requested subjects. At intermediate levels, menus are presented to the user, relating to the nodes that link to the current node.

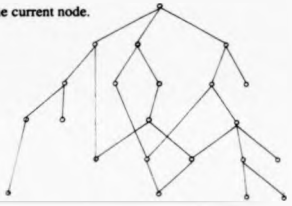


Figure 6.4 : representation of an information tree

(i) Top level entry

In the first development of the information system, access to the information system is from the top. The user is presented with some explanations about the use of the information system, before the top level menu is displayed :

```
-------------------------------------------------------
WES Menu Subject :  TOP LEVEL
   *  -  Display some general information about this subject
      -  WES
      -  Simulation
      -  Experimentation
      -  Prolog
      -  Glossary of terms
      -  Return to head of WESEF
      -  Go back to previous menu subject

-----------

Use the space bar to get to the required menu.
Press any other key to list the data or  [ESC] key to exit
-------------------------------------------------------
```

By moving the cursor (*), the user can select the appropriate topic of interest. This in turn will present the user with other menus until he reaches ground level where explanation will be presented to him. Such explanations may even hold a reference to a book, article or paper. This would even appear to be a reasonable way of keeping practitioners in touch with the research of academics. Ground level can be reached by asking for the option :

```
*  -  Display some general information about this subject
```

Within each level, the user has the option to return to the top of the information system or back to the previous menu. Access to a glossary of terms provides a useful service as many times the vocabulary used in the explanations may be too detailed for the knowledge of the user.

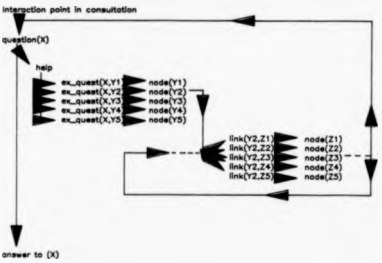
The main benefit of this system is that information can be held in a structured pattern that can be accessed by the user at all times. This system provides added potential to the task of educating users at the level of knowledge support. A lot more information can be stored for the amount of coding effort required. All that is required is a text file with the information and a node description to say whether it is ground or leaf level and a database of links to indicate what

other information relates to this node. At present, the system implements seven levels of explanation involving over 100 explanation topics (see submodule-E in the WES technical documentation manual).

The problem with this implementation is the loss of contextual information. Accessing from the top cause users to be lead astray from their original queries. They often investigate problems that are not directly relevant. And this is not necessarily due to new found interest but because they cannot work the tree structure to reach the area that covers their problem. To overcome this problem, it was considered to investigate entry into the system at question level rather than from the top.

(II) Question level entry

The user is now presented with a selection of topics that relate in some form to the question asked by the system that prompted the query in the first place. These topics represent possible entry points to the information system. They are selected through the use of a new explanation_question (ex_quest(question,node)) database, as suggested below.



This implementation was found to reduce users' time in finding the relevant part of the tree. A major benefit is that topics can be linked in extraordinary manners. For example, the normal tree structure might be as seen in figure 6.5.

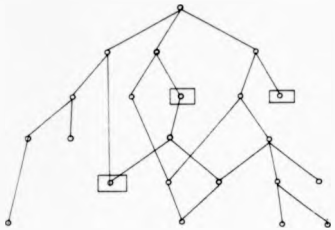


Figure 6.5 : The information tree with different entry points

The lines show normal node links. These links already permit movement in the tree. The new explanation_question database selects a number of topics (represented by []) as possible entry points into an explanation facility. Once entry has been selected, the natural control mechanism of the information system takes over.

(i) Limitations

The integrated information system unfortunately re-opened the problem of needing to adjust the explanation to the context. Also, the implementation described above of multiple entry points to the information system sometimes caused confusion about why a number of topics were suggested. The system appears to possess a good facility for queries about queries, once the information tree is engaged. But the first time response (ie a menu of possible entry points) from the system to a particular query is often unsatisfactory not to say confusing or too complex. It was considered that the amalgamation of the explanation facility with the information system may provide a more powerful means of helping and educating the user.

6.2.5 A Mixed Support Facility

The objective of this mixed facility is to keep the contextual feel by having the first time response from the "single explanation" facility, and link it into the information system for further

queries.

The system would support again three types of questions :

- HOW can the user provide this information,
- WHY does the system require this information,
- EXPLAIN something a lot more general to do with simulation theory.

HOW and WHY queries will use the mechanisms of the "single explanation" facility. First time support will be obtained from "explanation(X,['A'])" clauses. (See submodule_D for the explanation database in the WES technical documentation manual). If, when prompted with "ok ?", the user replies "no", the information system will be accessed at question(X) level (see figure 6.6). The information system will be accessed in the same way if an EXPLAIN request is presented. Appendix 13-d illustrates the use of this facility in a consultation.

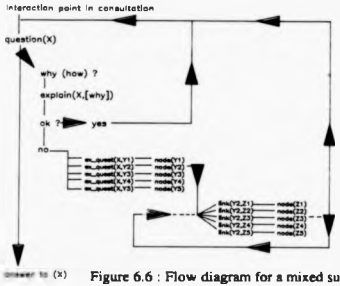


Figure 6.6 : Flow diagram for a mixed support facility

The benefits of this mixed facility are :

- first time responses are often all that is needed. They provide contextual tones to the explanations given.
- further explanation menus will present topics that have previously been introduced to the user in context during the first explanation.

- there is no longer any need to support "explanation(X,['explain'])" clauses or complex "select_query" databases.

- the user is provided with support about WES and simulation theory, and may direct his own investigation into the support he requires.

(i) Limitations

All the explanations are static. This means that regardless of the problem under consideration, they can always be retrieved. Only the process for retrieving them is dynamic. This mixed support facility does not address issues about checking the system's reasoning behaviour. This is addressed in the rule-trace explanation facility.

6.2.6 A Rule Trace Explanation facility

In the section 6.2.1.i on some general considerations, it was noted how different steps in the consultation required different types of explanation. It was felt that explanation of the reasoning of the system could be done through presentation of rules used. The question remained whether or not to consider static or dynamic interpretation. The work presented by Davis and Lenat (1982) underlined the complexity of dynamic interpretation. It was considered to see how much value could be gained from the static presentation of rules used. This requires a history recording mechanism. When a given rule is fired, a predicate of the form

assertz(arule(X,Y))

is executed. X will be the type of the rule and Y its number. The use of the clause "assertz" ensures a sequential storing of facts. If asked, the system can present each rule that has been used. A database is associated with this facility to provide a free translation of any rule that the user may wish to see.

This has proved satisfactory in order to allow the user to follow the reasoning pattern of WES, and was certainly adequate for debugging purposes. The presentation of rule translation on request rather than through a dynamic tree as in MYCIN was probably more appropriate because

of the number of rules that are used over and over again. It became apparent that a useful feature is the ability to turn the trace on or off for a certain class of rules. Generally only strategic rules are traced since only one is fired per decision and adequate information about the systems behaviour can be derived from this. For example :

Display rules used ? y.

The following rules were used during this consultation :
run_in_rule(3)
run_time_rule(1)
run_num_rule(2)
pred_rule(6)
eval_rule(2)

Present any rules ? y.
See run_in_rule(3) ? n.
See run_time_rule(1) ? y.

run_time_rule(1).

This rule was fired because WES had established that:
.....

Present another rule ? n

One of the main problems of traditional explanation facilities has been the lack of adequate "support knowledge" giving an explanation of the mechanism or associational links that explain why the conclusion section of a rule follows from its premise. This is overcome in the system as the user may ask for further explanation on presentation of a rule (figure 6.7). See appendix 13-b for an illustration and consult submodule-J in the WES technical documentation to view the rule-explanation database. The information system will be called up and entered at the rule level following the same principle as for question-level entry.

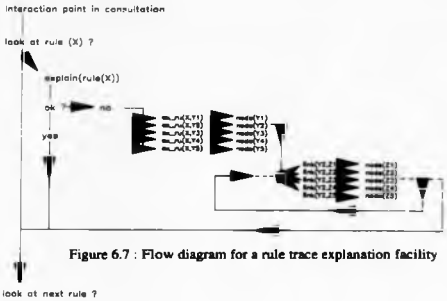


Figure 6.7 : Flow diagram for a rule trace explanation facility

6.2.7 Conclusion

The system now supports an explanation facility that provides different type of explanation depending on where the explanation request takes place in the consultation. Essentially, there is an explanation facility that integrates with an information system, and there is a rule trace facility that provides information about the reasoning process of the system.

The focus of the work carried out in this section was considering two main problems of typical explanation facilities. The first is the ability to adjust the explanation to the context in which the explanation is requested. This is not entirely overcome but is certainly addressed in the " two phase" mixed support facility where every first query has a contextual explanation that is related to the origin of the query ie. its question.

The second problem is that there is often a lack of adequate support knowledge to explain how certain rules come into being. This is addressed by linking the rule translation/presentation facility to the information system.

Technical details of the implementation of this facility may be found in the WES technical documentation manual in the main program and submodules D, E and J.

6.3 Teaching through Demonstration

6.3.1 Introduction

In section 6.2, the development of a facility that may enhance the user's understanding was presented as an explanation facility. One of the major limitations of that facility was the static nature of explanations. All explanations are model independent. For the user with little or no expertise, this may be too abstract and there may be a need to demonstrate the concepts and practices that are described and used in the explanation facility. For this purpose, the framework was devised for a demonstration facility. This facility covers two main tasks : demonstration of how to operate WES and demonstration of WES in operation.

Section 6.3.2 considers a tutorial facility that introduces the user to the WES system. Section 6.3.3 presents a facility that can demonstrate specific tasks performed by WES.

6.3.2 A tutorial facility to operate WES

The objective of this tutorial facility is to help the user understand how to operate WES. To this effect an example model is used : the die-shop model. The user is taken through the whole process of a consultation with WES : from introducing the model to selection of a task and appropriate experimentation and analysis.

The tutorial assumes that the interest of the user lies in how to answer questions from the system. In section 6.2.2, it was shown how every interaction goes through a question(X) clause. This makes it fairly easy to implement a tutorial session.

- First, a note must be made that the system is in tutorial mode : a fact "demo(1)" is asserted.

- Second, a new clause question(X) is introduced to cater for tutorial mode. This clause will open a text file (tutorial.WES) that holds the tutorial prompts. The user is told what is happening and what he must do. After every answer, he is told what has happened. The system asserts the answer held in the tutorial file.

```
question(X) :- demo(1),
        open(H,'tutorial.WES',r)
        tut_intro_(H,X),        ; introduce question
        question(X),            ; ask question
        tut_answer_(H,X),       ; retrieve tutorial answer
        tut_comment_(H,X),      ; comment what has been
        close(H),!.                     achieved
```

The user can be taken through carefully prepared tutorials that are to give him understanding about what to expect and how to operate WES.

- Third, the tutorial file must be prepared.

Example:

WES is asking you to input the names of the queues in the model. In this model there are several queues of interest : iamaq(11), iamaq(21), iamaq(31) and carq(41). Type in the name of the first one

---

Please input queue name : iamaq(11).

WES now knows that the name iamaq(11) refers to a queue in the model. It has stored this information as a fact.

6.3.3 Experimentation Tutorials

Experimentation tutorials have as objective to present to the user the nature of the tasks that can be investigated. The user is to be taken through the whole operations that WES undertakes. He can see what WES can do for him. It helps the user to specify the nature of the problem that he may be facing.

The implementation follows the principles of the tutorial facility presented in section 6.3.2.

- First, a note must be made of the experimental task that is to be demonstrated :

  - demo(2) for prediction,

  - demo(3) for evaluation,

  - demo(4) for comparison,

  - demo(5) for sensitivity analysis,

  - demo(6) for transient behaviour,

  - demo(7) for functional relations,

  - demo(8) for optimisation.

- Second, a new clause "question(X)" is introduced. It will open the appropriate tutorial file. The tutorial operations are the same for all tasks and questions. The user is given time to read the question. When he presses the space bar to continue, the answer is written on the screen and a comment appears. The user must press the space bar to continue to the next question.

```
question(X) :- demo(Y),
      tutfile_(Y), tuto(H),   ; open appropriate tutorial file
      quest(X), continue,     ; ask question
       repeat, skip (H,35),
         read(H,Z), Z == X,      ; retrieve tutorial answer
      read(H,Answer), write(Answer),
      assert(answer(X,[`Answer`])),
      wrt(H),
      retract(tuto(H)), close(H),!.

tut_file_(2) :- open(H,'pred.tut',r),assert(tuto(H)),!.
                                  ; open appropriate tutorial file
```

- Third, the file "X.tut" must be prepared. The user can then be taken step by step through the appropriate consultations and experimentation, using the die-shop model as an example. See submodule S in the WES technical documentation manual for an evaluation-tutorial file.

6.3.4 Limitations

The tutorial demonstrations provide a useful teaching method to enhance the user's understanding about experimentation and WES. Though it might be considered that the explanations are no longer static in that they are now model dependent, it is clear that they are not yet entirely dynamic. The user cannot attach his model for a tutorial.

The use of the demonstration facility by a number of experiment volunteers (see section 7.6) forced the refinement from a three-phase WES tutorial (introduction, question, comment) to a two-phase task tutorial (question, comment). This was found to be necessary as users could not endure another long tutorial.

6.4 Conclusion

The objective of the work presented in this chapter has been to investigate means of turning the support system developed in the previous chapters into a teacher. The concept of teaching was taken to be the ability to enhance human understanding. The postulate that was adopted is that it is possible to enhance human understanding through machine knowledge. Machine knowledge can be incorporated into the system and retrieved to teach the user. Two approaches to teaching were considered : explanation and demonstration.

Teaching through explanation was addressed using a facility that offered a mixed approach of traditional "how" and "why" questioning with a menu driven information system. At any interaction point during a consultation, the user is allowed to ask :

- how he can answer a given question,
- why the system requires this information,
- the system to explain a particular topic.

"How" and "why" queries obtain a first time response in order to ensure contextual relevance. Further explanations are then obtained from using the menu-driven information system. Entry to the information system can be at a variety of points depending on the original query level. It was found that this explanation facility added a particular tone to the use of simulation by allowing the user to ask "how", "why" and "explain" as well as the traditional "what-if" questions.

Understanding the reasoning of the system was made possible by the implementation of a rule trace facility that offered free English translations of rules if the user wanted to find out which rules applied. The facility was seen to be limited as it only offered static presentation of rules. However users could pursue their investigation using a facility that linked into the information system.

Teaching through demonstration addressed the issues of showing the user how to operate the system and how the system performs given tasks. The implementation was via the development of a tutorial mechanism that allowed any type of demonstration to be prepared in advance and presented to the user. An example program, the die-shop model, was used to illustrate all

points being made. Though not yet dynamic in that the user cannot attach his model to the tutorial system, the demonstration facility has removed some of the static nature of the explanation facility.

The outcome of this chapter is that it is possible to devise mechanisms that may enhance the user's understanding through machine knowledge in the domain of simulation experimental design and analysis. The actual assessment of the current usefulness of this facility is left to section 7.6.

End of Chapter

Chapter 7

Results and Considerations about the Use and Application of an Integrated Expert

7.1 Introduction

The preceding chapters have investigated the issues surrounding the development of a support system for the practice of simulation experimental design and analysis. A framework for an integrated system that advises on and controls the execution of simulation experiments has been formulated and a prototype system called WES has been implemented. The objective in this chapter is to report on the current and potential use of the developed system.

In line with expert system philosophy, the power of WES resides with the knowledge it embodies and its capability of making effective use of it. In section 7.2, the investigation of a number of retro-active studies highlights how WES was perceived as a suitable tool to address the problems presented and overcome some of the conventional "pitfalls" of simulation investigations. It is seen how these studies may have been improved with the use of WES. It is also seen how some of the default rules implemented were refined in order to produce satisfaction.

The second stage of the evaluation of the system's potential was to consider how robust the system currently is by linking it to some simulation models from industry. WES was linked to a data-driven simulation package called WITNESS and a number of models were considered. The data-driven characterisation made possible notions of automatic learning and understanding of simulation models by an advisory system: this is treated in section 7.3.

As a third move, the system's ability was tested against MBA students (section 7.4). It was seen how by making use of the optimisation facility, WES was able to perform comparably with these MBA students who had some simulation knowledge but little or no expertise. The results from this experiment prompted a study (section 7.5) into the design of more efficient "expert" rules by combining uncertain evidence and using "evidence" measures to derive decisions. The

MBA experiment was repeated successfully and rules were then applied to different models.

Finally in section 7.6, an experiment was conducted to assess the system's ability to help users with minimal or no simulation knowledge investigate and learn about simulation studies and concepts. The subjects of this experiment were presented with the same problem as the MBA students. It was seen how with the use of WES, they were able to achieve results comparable to students who had just completed a simulation course. The results tend to suggest that the system can help establish a positive awareness of simulation issues.

The chapter concludes that the framework developed in the previous chapters already supports usefully the design and analysis of simulation experiments, and is a valid framework for the support of this phase using artificial intelligence tools.

7.2 Some Retro-active Studies

7.2.1 Preliminary Analysis

Once work had begun on the development of the advisory system and an early framework had been proposed, a study was performed to assess the scope and potential usefulness of the proposed system to simulation experimentation. The original goal had been to consider how well the proposed system tied in with human practice and verify that it would mimic human behaviour to accomplish the same tasks. It was also hoped that lessons and guidance could be obtained about useful rules to implement.

The study reviewed 57 undergraduate and postgraduate student projects. The reasons for choosing these projects were threefold. The first reason was that students who had completed a simulation course were expected to present a certain level of expertise. The second reason was that as part of an academic exercise, students would have been expected to explain in detail the rationale behind all the steps that were carried out. This prevented the need to interpret or guess at some of the results that are not necessarily presented in simulation reports. The third reason was the simple matter of availability: they were easy to obtain.

(i) An early classification

Initially, these 57 projects were classified according to task. The idea was to see if the experimentation that was described could have been accomplished by WES. It was anticipated that at the very least, each investigation could be classified as a number of individual prediction tasks. Table 7-a outlines how the 57 projects were perceived as WES tasks.

The "other" classification became necessary because a number of projects had not gone beyond formal validation and the associated experimentation often involved dynamic changes in the configuration and reliance on the visual aspect for evaluation. Consequently, this type of experimentation could not be replicated even though the model could still be evaluated using WES.

Prediction : 8
Evaluation : 6
Comparison : 22
Sensitivity analysis : 6
Functional Relations : 1
Transient Behaviour : 1
Optimisation : 5

Other : 8

Table 7-a : Initial classification of student projects according to WES tasks.

During this classification exercise, a note was also made about the experimentation practice of the students. The reason for this was to establish the amount of knowledge required for the advisor to perform at the same level of expertise. During this observation phase, it was noted that student knowledge encompassed:

- awareness of run-in problems;
- awareness that different configurations may require different run-in periods;
- consideration of a number of different configurations;
- awareness of the need for replications (and variation of seeds),
- considerations about the length of runs;
- knowledge of variance reduction techniques (such as antithetics);
- consideration of evaluation criteria;
- formal analysis of results;
- use of different experimental frames (long runs with multiple sampling or multiple shorter runs with single sampling);

It was decided that the Advisor should be able to display the same amount of knowledge. Refinement of rules and structure of WES subsequently focussed on achieving this objective.

The outcome of this initial study was a positive indication that the proposed system would tie in with the practice perceived from these projects. Many of the necessary features, as presented in chapter 4, were identified, at this stage, in order to permit implementation of design

and result collection mechanisms.

(ii) A detailed classification

Once a more definitive version of the Advisor had been implemented, a more formal study was undertaken to investigate the actual quality of these projects. Although in the earlier classification, the opportunity had been taken to consider the knowledge displayed, it had not been assessed whether all the studies considered the issues involved. It was anticipated that since the Advisor now possessed this knowledge it would be able to fare equally well, if not better.

The study highlighted the fact that despite the level of knowledge that was globally attained many studies were weak in some aspect of experimentation. Some of the more pertinent findings of the study are presented below, whilst the detailed classification may be found in appendix 9-a. (Tables prefixed with an "a" may be found in this appendix).

In terms of simulation practice, it was noted that nearly 80% of simulation studies investigate formally 5 or less configurations and only 10% of the time are more than 10 different configurations analysed. (table a7-1).

The main criterion for measuring performance was the time-in-the-system of entities (72%). This was not a surprising result since the simulation package used permits easy recording of these measures. Queue occupancies which are also available are not so popular (3%). (See table a7-2) Most of the students preferred the multiple short run frame (70%). (See table a7-3).

In terms of formal analysis, most students relied on the use of confidence intervals to make their decisions rather than formal hypothesis testing. Nearly a fifth of the studies had no formal analysis (table a7-4). In terms of other malpractice, (table a7-5) at least 47% of the students ignored the issues of steady states and how to achieve one if required. And of those that did consider the problems of run-in problems, only 10% acknowledged the fact that different configurations may require different run in periods (table a7-6).

If a sample of 10 is considered to be the absolute minimum size for a representative group, then only 44% of the studies satisfy this requirement (see table a7-7).

Considerations of the use of variances reduction techniques were ignored by at least 44% of the students (table a7-8).

In principle, WES should have been able to overcome all the weaknesses since it had been given knowledge in all these areas. As validation of this reasoning a number of retro-active studies were performed. From the classification found, several projects were selected to illustrate WES' ability in each of the expert tasks (see table 7.b). These retro-active studies were also aimed at the verification of WES. Additionally, parts of other projects were replicated using WES to test secondary level rules, such as a CYCLE SHOP simulation used for run-in time investigation.

	LAUNDRETTE	Evaluation
	MARKET STALL	Comparison
	RUBBISH DUMP	Sensitivity Analysis
	AIRPORT	Functional Relations
*	PETROL STATION	Transient Behaviour
	IMMIGRATION OFFICE	Optimisation

Table 7-b : List of formal studies and the tasks they address.
* not linked to WES

The retro-active study entailed using WES to carry out the experimentation described in the project report. All but one of the models were directly linked to WES. Each of these studies was successfully replicated in terms of the experimentation : WES cannot put any social-economic meaning to any of the results produced. The defaults rules that WES currently held performed adequately if not always most efficiently. A number of further refinements were suggested from these studies mainly in the area of result presentation.

Following from these studies, a number of benefits came to light. They are presented here in anecdotal form. The reason for this is that no real significance can be attributed to them apart from indications of added usefulness provided by WES. The studies presented cover four topics :

- correct execution of designed experiments;
- steady state issues;
- replication;
- closed view analysis.

7.2.2 Correct Execution of Designed Experiments

(I) The "bad" recording of results.

From a basic practical point of view, the controller offers perfect recording. It does not make mistakes in transcribing results from the simulation to the analysis phase. Although a trivial point, this mistake has been witnessed several times. In a particular project concerning the simulation of a market stall, it was discovered that the comparison of three different configurations varying the number of servers only, revealed that two servers was slower than one server, but three servers was faster than one server. Despite correct design, this unfortunate result was then interpreted from a behavioural point of view. The student who performed this study suggested that the servers would *"take longer over jobs because there are others to do the same job"!!* WES was presented with this problem using its "comparison" rules. The results from WES produced a natural decline in customer waiting time. It appeared when the experiment was re-performed manually that these erroneous recording were obtained by reading the maximum time in the system as opposed to the average.

(II) The "wrong" set up.

Again from a very practical point of view, it was noticed during a functional relations exercise that the student-analyst of a study was using the wrong configuration. The model in question simulated airside activity at an airport terminal. Though the problem had been about budget allocation concerning the number of tugs, fuel tankers, steps and baggage loaders, the student had wanted to investigate the relationship between the arrival rates of aircrafts and their servicing

times. WES initially used this model to test its ability to address "functional relations" problem. On comparison of WES' results with the student's, a discrepancy was noticed in servicing times. This was explained when a different configuration relating to the number of tugs, fuel tankers, steps and baggage loaders was considered. It transpired that the author had not reset these parameter to the values that the report suggested.

7.2.3 The steady state issues

Traditionally a difficult part of every study, the review of students projects reflected this and indicated a number of issues that derive from addressing the steady state problem.

(i) No run-in time consideration

In a number of cases (47%), it was detected that no consideration was given at all to whether or not the system may be terminating. However, this was sometimes due to more subtle understanding of the problem. One example was the simulation model of a motorway petrol station. The author of this study had wanted to investigate the "rush hour" effect. The rush hour was generated by different arrival rates from normal traffic times. For this the author had compared the two traffic flow during two hour periods. However there is no evidence in the study that he considered running the model normally until rush hour time, and then changing the arrival rates and begin monitoring. Although changing configuration parameters during the actual running of the model is beyond the current implementation of WES, it was possible by use of the transient behaviour rules and the stand-alone version of the system to recommend that the model be run until the beginning of the rush hour time when monitoring should begin for that period until end of rush hour time.

(ii) Run-in time rules

a. Default Rules

Detecting appropriate run-in time is certainly one of the interesting features of WES, as very few packages offer any support in this respect. As was mentioned in chapter 3, WES has the ability to set up investigatory experiments to detect appropriate run-in periods. This is implemented by a monitoring system on the controller side. Initially the average total utilisation of "activities" was used to monitor transient effects. The assumption was that once this average total utilisation begins to stabilise then the system is in balance. A number of rules were implemented to select the appropriate cut-off point. Heuristics adopted were to consider the last three observations (X(t), X(t-1) and X(t-2)).

- all three observations must be within range
  - $|X(t) - X(t-1)| < Range$
  - $|X(t-1) - X(t-2)| < Range$
  - $|X(t) - X(t-2)| < Range$

- not all increasing: not $(X(t) > X(t-1) > X(t-2))$
- not all decreasing: not $(X(t) < X(t-1) < X(t-2))$
- all strictly positive $X(t) > 0$, $X(t-1) > 0$, $X(t-2) > 0$

The consequence of these rules is that once a local maximum or a local minimum has been reached, then this is considered to be a suitable run-in period. Heuristically, the time between observations is set equal to 100, but is only a parameter and can be changed. One might consider rules for determining the appropriate recording interval as a function of the average time in the system of entities.

b. Refinement

On many models, this proved to be an adequate measure. However, it did happen that the total average utilisation could be affected by early bottlenecks in the system thereby causing utilisations to stabilise too soon. An alternative was to consider the total average time-in-the-system of entities. This in fact proved much more effective in ensuring that all processes had been used before the cut-off point was selected for the run-in time. The rules used were compared to a

number of approaches that conventionally ran the model for an extremely large number of time units and then considered the cut-off point (see Pritsker, 1984) . Generally, where students had performed formal investigations in this way into the run-in time issues, it was found that they opted for a 20% longer transient period than WES. Observation of graphs suggested that there was no underlying mathematical rules for this, but that a heuristic suggests caution and recommends the increase to be sure that the transient period will be over all the time. WES on the other hand tends to cut-off early. Although it would be easy to add this heuristic to WES rule-base, as it was not practised all the time it was left out.

This example highlights the problems of knowledge engineering when there is inconsistency in the practice of experts.

c. Limitations

Evidently more rules need to be added to achieve more appropriate cut off points according to each models characteristics. Further illustration of this and indications that WES rules cannot solve every problem yet were given in a particular simulation study of a cycle shop, where a student had considered five different configurations. He had run the model for 25000 time units with each configuration. WES was also asked to consider these five configurations. Results as shown in table 7-c revealed that initially sampling (as explained above) every 100 time units, WES came up with much shorter run in times. This was due to the fact that WES could not know that 100 time units was insufficient time span. (Initially, some clock advances were over 200 time units !!). When the same sampling interval was chosen as done by the student (every 1250), WES results were much closer (see table 7-c).

(iii) The overkill

Although it was found that 37% of the students did display awareness about the need for run-in time considerations, this did not mean that they always found sensible durations for the transient period. It was found that students often went for the extra large run-in time when in doubt. An

Configuration number:	1	2	3	4	5
Student cut-off point:	6250	7500	6250	7500	10000
WES cut-off point:	800	400	400	400	400
WES adjusted recording:	6250	6250	6250	6250	6250

Table 7-c : Comparison of WES and student cut off points.

example was provided during the analysis of the market stall. The student set each run-in time to 6000 time units whilst WES identified 300 time units as being sufficient (considering the average time-in-the-system of entities never exceeded 51 time units, 300 seemed realistic enough). The student then proceeded to run experiments of 10000 time units between observations, whilst WES considered 300 to be adequate. For the same number of sets of observations (20), the student ran 212 000 time units in two long antithetical runs, whilst WES selected 12 000 time units in 20 short antithetical runs. As for the accuracy of results, the confidence intervals were never more than 10% larger. These results open the issue that surrounds the efficiency of solutions. If computer time is valuable, clearly there is some value in more expedient solutions as offered here in this example by WES.

(iv) The single run-in time for all configurations

Another common problem is that when a run-in time investigation is actually presented, the result is often used for all variations of the configuration. This is not necessarily correct as was exemplified in the functional relations problem about airside activity at an airport terminal. Indeed as the inter-arrival rate of aircrafts increased so did the system take longer to warm up. This student chose as a common run-in time 65 time units run-in time period for all configurations (even though the mean time-in-the-system was seen to vary from 104 to 239). WES varied the run-in times from 400 to 1000.

7.2.4 The "closed" view analysis

A danger of simulation studies is the mismanagement of data. Simulation runs produce much data and the analyst is often forced to restrict his attention to what seems important to him. This was exemplified in a sensitivity analysis study performed by WES on a simulation model of a large municipal rubbish dump. In this case, the student who performed the study was considering which factors would most affect his proposed dump. The student performed some sensitivity analysis on the arrival rates of lorries and cars and declared himself happy with the configuration he had found even though the figures of his own study revealed that a change of a parameter relating to lorries processing could reduce their time in the system by 25%.

By using WES, the author performed the study with the same criteria and was informed about the significance of the parameter on the performance of the model. WES offers the ability to analyse all the data and bring to the user's attention extraordinary facts that may be lost among all the other results.

7.2.5 Benefits from WES

The experience related in this section is only designed to illustrate the benefits that may be derived from a system such as WES. There are indications that WES may offer efficiency of solution, overcome errors in the recoding of results and the execution of experiments, ensure that variance reduction techniques are employed, and keep an objective view on all the system performance measures.

Much refinement is probably needed to obtain satisfaction at a greater level of expertise, however it is found that the system works adequately at the level of simulation student projects.

7.3 Application of WES to a Data-Driven Simulation Package

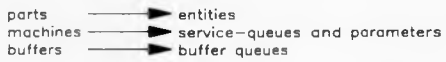
In section 7.2, a number of models were used to help investigate the current practical use of WES. Though they provided critical insight into its potential, it was felt that the application of WES to industry-developed problems might reveal more about its ability. At the same time, if WES could be transported away from a conventional FORTRAN-based simulation system, more may be learnt about its robustness. WITNESS was chosen as it offered added potential to a tool such as WES. WITNESS is data-driven : ie models are programmed as data. WITNESS "is designed with the inexperienced user in mind (ISTEL, 1987) ". ISTEL also quote " Previous simulation techniques required specialist operational research skills, but now managers can use the package to quickly and easily explore options, identify key factors and make informal and therefore better decisions". Personal experience has revealed the power of WITNESS in model building, but also accentuated the concern expressed in chapter 1 that misuse of simulation will be made easier to occur. For these reasons and because ISTEL donated a selection of models for evaluation, it was decided to link WES to WITNESS models.

7.3.1 WES and WITNESS : some general considerations

The objective of this short section is not to present an introduction to simulation using WITNESS, but rather to consider how some of its components and characteristics may be used with WES.

(i) Consideration of Simulation Components

WES associates with models : entities, parameters, queues. WITNESS is directed towards manufacturing applications and the components of its models are so expressed. For example, it has parts, machines and buffers. It is necessary to translate these component concepts into the language of WES. There are a great number of possible components in WITNESS simulation models, but attention will be restricted to :



Entities, service queues and parameters are already understood by WES. The concept of a buffer queue is new and will be introduced to show how WES can incorporate new component types.

(ii) Input-Output Characteristics

WITNESS is a powerful package in terms of statistical recordings: simulation studies produce detailed quantitative analysis of individual runs that are automatically generated in report files. There is information associated with each of the components in the models. With the current implementation of WES, it has been necessary to restrict the current information concern to performance measures that WES currently understands. These are :

- for entities : time-in-the-system;
- for service queues : utilisation.

However to illustrate that it is not too difficult to expand the capability of WES to considering other performance measures, the occupancy of buffer queues are also analysed.

7.3.2 Linking WES to WITNESS

The requirements for an integrated system were identified in chapter 5 as the ability to:

- communicate the names of the components in the model (model knowledge);
- initialise experiments by setting :
    - parameter values;
    - run length and run-in times;
    - variation of seeds;

(experimentation data: input from Advisor to Controller)

- collect results to transfer to PROLOG. (experimentation data: output from Controller to Advisor)

This may seen as the transfer of experimentation data characteristics and model knowledge.

(I) Transfer of Experimentation Data Characteristics

The MICROVISION implementation required the existence of a controller that is implemented as a superstructure to simulation models. In this case, it was not possible to develop such a controller, and an alternative was found that illustrates the robustness of the framework to allow "pro-controller" devices.

(a) Input from WES to WITNESS

In the parallel link, communication was via assembler serial interface. In the MICROVISION sequential link operations were conducted at operating system level through shared data files. Because it was not possible to change and adapt the package, the link to WITNESS considers sequential linking at operating system level but conducts experimentation by redirecting keyboard input from a file. These changes for WES are conceptually easy. Instead of preparing as in figure 7.1, a file of input values for the controller to use in the execution of experiments, WES prepares a file of instructions (figure 7.2) which the WITNESS package executes.

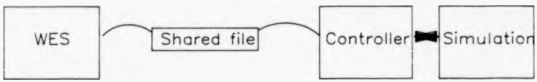


Figure 7.1 : Shared data file for WES-MICROVISION communications

Figure 7.2 : Instruction file for WES-WITNESS communications

Since keyboard input is now simulated, the impression of an "expert" machine simulationist is deepened. Not only does WES mimic the experimentation process as identified in chapter 4, it now executes in the same form as a human expert. Also WES learning of these activities resembles more closely the expert system knowledge acquisition process. The author was compelled to learn these techniques first and then teach WES to perform the same tasks. WES is given the ability to :

- select a model;
- change parameters;
- select the duration of runs;
- select seeds and antithetic mode;
- specify what to monitor.

An example instruction file is given in figure 7.3.

```
-----------------------------------------
NO,FACTORY                  : select the model
GO                      : start the simulation
SELECT,TRUCK,Q,2        : change parameters
BATCH,DURATION,1440             : select duration of run
MODE,A                  : select seed and antithetic mode
REPORTS                 : specify what to monitor
OUTPUT,FILE,FACTORY             : prepare result file
;
FACTORY,N
 Q                      : quit simulation
-----------------------------------------
```

Figure 7.3 : Typical instruction file.

The implementation of this "pro-controller" device does not affect the rules or the consultation process. A couple of alternative PROLOG clauses are only required to prepare this instruction

file instead of the shared data file of previously. It was found useful to develop a fact "prleg(X)" to designate the simulation package that was being used. A check for the appropriate clause could be made by detecting this fact.

In terms of computer efficiency, linking using a keyboard substitute file is much slower than the direct link, but offers more insight into the processes that are carried out.

The implementation of the sequential operations for an integrated WITNESS support system may be found in appendix 10-d.

(b) Output from WITNES to WES

Results are collected from the report files automatically generated by WITNESS (see appendix 10-b). Since these files all have standard output, it is possible to get WES to scan these files to pick up the performance values that are required. As identified above, these will now be about:

- time-in-the-system of entities;
- utilisation of service queues;
- occupancy of buffer queues.

For example, in figure 7.4, a part of a report file is presented that holds results about the utilisation of machines. WES may examine this file and translate the information into its own format.

```
Page 2                    FACTORY REPORT                          Time 1440

MACHINE          NO OPS   %IDLE    %BUSY   %BLOCK   %SETUP   %DOWN   %WAIT
-------          ------   -----    -----   ------   ------   -----   -----
TRUCK(1)            761    2.91    97.09     0.00     0.00    0.00    0.00
TRUCK(2)            756    4.05    95.95     0.00     0.00    0.00    0.00
```

Figure 7.4 : Part of an example report file

In this case, WES would pick up the utilisation of machines TRUCK(1) and TRUCK(2). These values would be stored as facts :

wfac(900,E,[truck(1)],R,97)

where E would be the experiment number

R the run number

The consultation could then proceed as normal. The implementation only requires different clauses for result collection : these may be seen in the WES technical documentation manual (module W).

(ii) Transfer of Model Knowledge

In the MICROVISION implementation, it was considered useful to force the user to declare the important components of the model. In the retro-active studies however, the author was forced to perform this task, thereby indisputably acquiring some understanding about the characteristics of the system. The task is tedious and its effectiveness depends on the author/user to perform this task correctly. WITNESS offers a more powerful approach. The description of a WITNESS model is held in data form (see appendix 10-a). This form is readable by the human analyst. Another user may read this file to get some early insight into the system. Illustration of this is seen below.

The WITNESS file provides a declaration routine of the components of the model :

```
DEFINE
    Part : A,Variable attributes;
    Machine : TRUCK,1,Single,9.0;
    Buffer :  A_BUF,1,50;
```

To the human analyst, this definition translates as :

- there are parts, called A in the modelled system (Conventionally, these would be called entities).

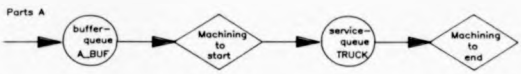
- there is a machine called TRUCK. This has two implications :

    - first there is a parameter that controlled the number of such machines (in this case 1);

- second, this machine may be designated by a service-queue called TRUCK.

- there is a buffer called A _BUF. This can be viewed as a buffer queue.

From this simple definition the following understanding may be derived :



If WES could read this description file and translate it for the user, this would provide evidence of automatic understanding by the system of the model. This was achieved by developing some new clauses in the model understanding phase of the WES system and this is presented in the next section.

7.3.3 Automatic Model Understanding

WES may be taught to read the definition data-files. This is achieved by getting WES to scan these files for keywords such as "DEFINE", "Part", "Machine", "Buffer" and then analysing the appropriate lines to record names and possibly some characteristics.

From the consultation sequence, the user may specify a model to be investigated of which he knows very little. WES picks up this name and opens the corresponding data-files. It then proceeds to establish in turn :

- the entities in the system;

- the parameters and their default values;

- the service queues;

- the buffer queues.

The "discovery" experience is shared with the user as seen by a screen dump in figure 7.5.

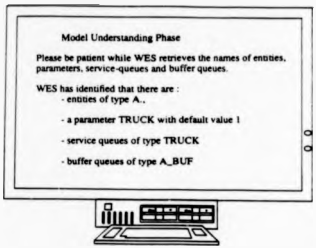


Figure 7.5 : Screen showing the WES discovery process of WITNESS models

One of the problems encountered was the mixing of lower and upper case characters. WITNESS only accepts upper case whereas PROLOG only accepts lower case. A translation mechanism was required. The implementation may be found in the WES technical documentation manual (module W).

As before, WES stores these facts in a form suitable for its own purpose. Since it was decided to add buffer queues to the knowledge of WES, new facts of the form bname(Number,[Name],Good,Bad,Noted) were created. All these facts may then be saved for further consultations and this "discovery" operation will be omitted.

This achieves the same outcome as the parallel link. However, the implications of this "understanding" process is that theoretically WES should be able to assimilate all the model knowledge that the simulation has been given by the analyst. This would necessarily require some added support structure to deal with the different knowledge representations and further rules may be required. Considerations of this possibility is provided in section 7.3.5. An illustration of automatic learning is examined in the next section.

7.3.4 Automatic Learning : evaluation of WITNESS models

In the previous section, it was seen how a user could specify a model to be investigated of which he knows little or nothing. WES automatically "understands" about the components of the system and shares this experience with the user. The next stage of model understanding would be to see how well the simulated system performs under its default configuration. For this, it was found that selecting the "evaluation" task from WES' repertoire provided insight into the simulated system since it is possible for a user to request a detailed evaluation of the whole system and proceed through the consultation by requesting default options if no knowledge about the system is known.

Through the normal consultation process, the model is run and results collected in order to evaluate the system. Typically, the user is presented with the output of figure 7.6 :

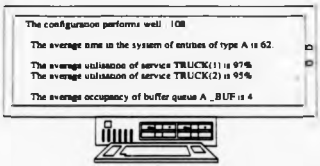


Figure 7.6 : screen output from evaluation of WITNESS models

Within a couple of minutes of keyboard time, a user may learn about the model that he is considering. This provides a powerful insight into the system that is being investigated. The user is provided with a feeling for the typical values of the performance measures in the system. The quantification value of the system performance is only really useful when other configurations are considered.

WES itself may be credited with automatic learning, since the user has taught it nothing about the knowledge it now possesses about the model. Several models that had been donated by Istel were investigated in this way.

7.3.5 Increasing WES Expertise

In the previous chapters, a framework was presented for an integrated advisory system which it was argued could support the addition of extra expertise. Quite obviously a number of changes were necessary to the WES system in order to permit linking to WITNESS (see module W in the WES technical documentation manual). The opportunity to test the robustness of the framework further in this respect was taken and a number of extra features were added.

The power of the system is reflected by the expertise it incorporates. This expertise is a function of the knowledge that is stored and the system's ability to support this knowledge. In other words, the performance of the system depends on the structure it has to use the knowledge that may be incorporated. For example, WES might have sophisticated rules about when to use a given experimental frame, but if it does not support the execution of this frame, then this expertise is worthless to an integrated system. Alternatively, as suggested at the end of section 7.3.3., WES may not have a knowledge representation or a structure to learn about the different aspects of a model. Even if the simulation package has good analysis facilities, but WES cannot make use of this, then it is lost. In the following, it is precisely the increase of the system support structures that is investigated.

(1) Steady State Determination

The FORTRAN controller for MICROVISION models sampled every X number of time units and performed a series of checks in order to decide whether a steady state can be assumed to have been reached. As soon as it has, the run-in time investigation is halted.

Since there is no formal "controller", a different approach was required for WES and the opportunity was taken to investigate a different approach. A long run with multiple sets of observations which are then analysed in order to determine the appropriate cut off point. (figure 7.7)

This is implemented by getting WES to design a long run with multiple sets of observations. The defaults are to sample 20 times every 100 time units. The same rules as before were used initially, but this frame allows further checks in time in line with student practice as recorded in

section 7.2.3.

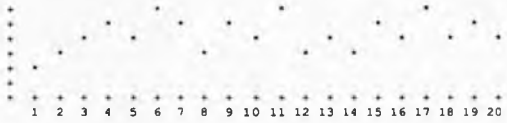


Figure 7.7 : plot of collected samples over a long run.

(ii) Additional Experimental Frames

Adding a new experimental frame is not too difficult since it only appears in the rule-base. The ease of execution of these frames, however, is dependent on the nature of the "controller" implementation. In the case of a formal implementation according to the design of chapter 3 as a superstructure, it may require much effort. In the case of instruction files, it has been fairly easy. Users of WITNESS models are now offered three experimental frames:

- multiple short runs with single set of observations,(figure 7-8a)
- long runs with multiple sequential sets of observations,(figure 7-8b)
- long runs with multiple sets of observations separated by transient periods. (figure 7-8c).

Figure 7.8a

Figure 7-8b : Frame 2

Figure 7-8c : Frame 3

Details of the implementation of these frames may be found in the WES technical reference manual (module W).

(iii) Additional Performance Measures

Adding a new performance measure requires some effort since it has to be integrated in every stage of the consultation process. However, it was found not to represent any real difficulty as all that is required was to emulate the structure supporting the other performance measures (such as entities or service queues). Rules are required to analyse the output from these performance measures but again the structure already exist to support the implementation of these rules.

In this exercise, queue occupancies were added to WES and the addition of extra rules permitted refinement of WES analysis capabilities.

(iv) Use of Package-Dependent Rules

The above has illustrated how it has been possible to increase the expertise of WES by extending its structure to allow the implementation of more rules. However, much of this expertise is to do with the practice of WITNESS simulation experimentation. The WITNESS package permits a number of features that may not be readily available on every simulation package. Therefore, much of the added expertise required may be kept separate from the main system and consulted only when WITNESS models are considered.

The interface was changed to allow this feature and a file called WITEXP was created to hold all the rules that are WITNESS-specific. (See appendix 11-a).

7.3.6 Results and Some Concluding Considerations

The exercise of linking WES to WITNESS has provided some significant conclusions.

(i) - WES does stand up to transfer to another simulation package;

(ii) - Linking WES to a data-driven package offers added potential to simulation studies. It was seen how a user with little or no knowledge about a model could select it for consultation with WES. WES would automatically learn about its components, its performance under the default configuration, and ability to reach a steady state.

(iii) - The use of package-dependent expertise provided the potential of taking full advantage of the existing features of a simulation package.

(iv) - The framework of WES is sound and can be expanded to support further expertise. It has been possible to illustrate a couple of the concepts that were mentioned in the previous chapters but not implemented in the MICROVISION implementation :

- additional frames,
- package-specific expertise,
- taking advantage of the power of individual packages.

(v) - The interface of WES was altered to cater for linking to alternative packages.

(vi) - Potentially WES can link to any type of package provided :

- a translation phase can be identified;
- one can develop a virtual controller;
- input-output mechanisms can be identified.

Appendix 13-a provides an example consultation using a WITNESS model.

7.4 WES versus MBA students

7.4.1 Problem description and MBA results

(i) Motivation

In section 7.2, a number of retro-active studies were performed in order to establish the suitability and appropriateness of using WES to overcome some typical problems of experimentation. In section 7.3, the robustness of WES was further investigated by applying it to a different modelling representation. It was found to offer great promise in the domain of automatic learning and understanding. In this section, a quantifiable and qualifiable assessment of WES current performance is sought after. To this end, WES is presented with an "optimisation " problem that confronted MBA students. It was anticipated that this experiment would provide some interesting considerations about the usefulness and potential of WES and indications of the system's limitations.

(ii) The ARGON Problem

As part of an MBA management science course, students were presented with a simulation assessment, that involved the use of a verified and validated model. With the use of this model, students were asked to determine suitable staffing levels for a company (ARGON) retail store. The full description of the problem is provided in appendix 12-a. Apriori, the solution procedure would involve the variations of the parameters relating to the number of servers at different service areas. The components in the system can be identified as :

entities : customers of type cheq
customers of type cash
customers of type cred
customers of type view
customers of type jewl
customers of type retn

```
parameters : order_servers
             cheque_servers
             cash_servers
             credit_servers
             collect_servers

service queues : orders_qu
                 cheque_qu
                 cash_qu
                 credit_qu
                 collect_qu
                 jewel_qu
```

Students were provided with a means of retrieving mean figures about the time in the system of entities in addition to on screen inspection of queuing activities.

The starting configuration is unstable and hence the "optimal" solution will involve variations of a number of the starting values. It was decided that the experiment would prove conclusive if WES could provide a configuration that matched up with the consensus of the class.

(iii) MBA Results

The results of 39 MBA students were collected and analysed. The full study may be found in appendix 9-b but the most significant points of the analysis of their results are presented below.

Configuration results are presented as quintets (A,B,C,D,E) where :

A is the number of servers at the order counter
B is the number of servers at the cheque till
C is the number of servers at the cash till
D is the number of servers at the credit till
E is the number of servers at the collection counter

There is no absolute correct configuration as parameter may range between bounds. Outside these bounds however the configuration will be considered "wrong".

In total, nine different possible "optimal" configurations were put forward by the students of which four are dominant (table 8-a) and one is selected by 36% of the class.

More representative was thought to consider each parameter value in turn and identify the most "popular value". Table 8-b reveal that for each parameter there is a value chosen by more than 75% of the class.

Or	Ch	Ca	Cr	Co	selected	by
4	2	2	2	3	:	1
4	3	2	2	4	:	6
5	3	2	2	3	:	6
5	3	2	2	4	:	14
5	3	2	3	4	:	5
5	3	3	3	4	:	1
5	3	3	3	5	:	1
6	3	4	3	4	:	1
6	3	2	2	4	:	2

Table 8-a : Different "optimal" configurations

Order :
4 : 7
5 : 29 76%
6 : 3

Cheque :
2 : 1
3 : 38 98%

Cash :
2 : 35 90%
3 : 3
4 : 1

Credit :
2 : 31 79%
3 : 8

Collect :
3 : 6
4 : 32 82%
5 : 1

Table 8-b : Most popular parameter values.

This would suggest that the best selection of values is : 5 3 2 2 4.

Some consideration was given to the quality of the MBA students' experimentation. The number of alternatives seemed normally distributed around a mean of 7 (table a8-2 in appendix 9-b). From the point of view of the experimental design, only a handful failed to take a sample larger than 10, and 82% took a sample larger than 15, which constitutes a satisfactory sample size

for a t-test (table a8-3). It was also noted that 87% used confidence intervals to analyse their results (table a8-4). The method for sampling was equally divided between long and short runs (table a8-5). The criterion for 90% of them was to use time-in-the-system of entities (table a8-6). This all translated into a feeling that the consensus of the class should be fairly accurate.

7.4.2 WES and the ARGON problem

As explained in the previous section, WES was given the ARGON problem. The model and its components (entities, parameters, queues) were linked up to WES in the standard way and WES was set the task of optimising the model. All the default values of WES were used so as to mimic the use of WES in the hands of a non-expert simulation user. Since the MBA students used as criterion the time-in-the-system of entities, the first experiment considered this criterion. Thereafter, it was decided to try out the other two main criteria open to the user : a general default criterion and consideration of maximisation of queue utilisation.

(i) MBA versus WES' minimisation of total time-in-the-system

This is in line with the criteria open to MBA students. The implementation of this criteria considers minimisation of total average time-in-the-system of entities.

The design of experiments was similar to all three experiments for each criterion used :

- the system was viewed as terminating (hence no run-in time) over the period of 14400 time units, which represents the four hour period of a busy Saturday morning in seconds.
- each configuration was to be run 14 times with the use of antithetic random numbers streams.

The design of the experiments was automatically set up by WES after consultation with the user (the author) and its rule-base.

In this case the results were the configuration 5 3 2 3 4 (see figure 8-c), which corresponds to one of the most dominant configurations and matches up 4 out of 5 of the most frequent

parameter values of the MBA results. The value of the fifth parameter is still acceptable, and in some sense the most important objective of finding a stable configuration has been successfully executed.

This result was seen as very encouraging as it indicated that the framework of the system could support a lengthy experimentation process and that the rules implemented, though not necessarily expert, could address the task of optimisation. In this respect, it was concluded that the system could perform as well as MBA students who are seen as simulation users who have knowledge but little expertise.

The rules cannot be considered yet as having great intelligence as it takes WES 12 configurations to decide upon the optimal configuration. Note there are several configurations that have the same performance. However, WES interprets this as the change that occurred in these later configurations did not improve the performance of the system and so the first of these configurations is chosen. (This relates to notions of indifference zones as reviewed for example by Kleijnen (1977)). For example configurations 6 and 7 differ by the number of servers for cash (3 or 4). The fact that they produce the same performance for the system suggests that 3 is sufficient since 4 does not improve the system. An underlying assumption is that fewer resources are better than more.

Ex numb	Perf	order	cheque	cash	credit	collect	changed
1	73	2	1	2	3	4	
2	86	3	1	2	3	4	1
3	92	4	1	2	3	4	1
4	93	5	1	2	3	4	1
5	95	5	2	2	3	4	2
6	96	5	3	2	3	4	2
7	96	5	4	2	3	4	2
8	96	5	3	3	3	4	3
9	96	5	3	2	4	4	4
10	95	5	3	2	2	4	4
11	96	5	3	2	3	5	5
12	94	5	3	2	3	3	5

Table 8-c : Output from WES about system investigation using minimisation of the time in the system of entities. (Ex numb is experiment number, Perf is system performance, order..collect are the staffing levels used at the different desks, and changed refers to the parameter changed from the previous experiment).

(ii) MBA versus WES' maximisation of system utilisation

In this case, WES considered the average total utilisation of the system's service queues. This facility was not open to MBA students. At best, they could only observe queue build-ups on screen. This criterion is supposed to make maximum use of staff inside a stable configuration. The following result was obtained after 14 configurations (see table 8-d) : 4 1 2 2 3. The configuration is unfortunately unstable. This resulted because the traffic intensity of that service was only fractionally greater than 1. This meant that the inclusion of an extra server caused the overall average utilisation to drop too significantly to remain acceptable to WES.

It became apparent that this criterion could only work with a strict control on the stability of the system under a given configuration. A monitoring device that checked the time-in-the-system of entities was implemented for WITNESS models. The associated rule simply stated that if the time-in-the-system of entities is continuously increasing then the configuration is imbalanced. Should this happen, the configuration is reported as unstable and WES proceeds to the ...xt configuration (module W in the WES technical documentation).

Ex numb	Perf	order	cheque	cash	credit	collect	changed
1	26	2	1	2	3	4	
2	38	3	1	2	3	4	1
3	40	4	1	2	3	4	1
4	34	5	1	2	3	4	1
5	30	4	2	2	3	4	2
6	32	4	1	3	3	4	3
7	39	4	1	2	4	4	4
8	44	4	1	2	2	4	4
9	42	4	1	2	2	5	5
10	49	4	1	2	2	3	5
11	58	4	1	2	2	2	5
12	53	5	1	2	2	2	1
13	49	4	2	2	2	2	2
14	52	4	1	3	2	2	3

Table 8-d : Output from WES about system investigation using maxisation of the service utilisation.

(iii) MBA versus WES' default criteria

The default criteria considers the time-in-the-system of entities and the utilisation of service. A formula is adopted to reward high utilisation of service but penalise excessive time-in-the-system:

100 + total average utilisation - scaled (1-100) average total time-in-the-system

This criteria gave the configuration 4 2 2 2 3 as the best. All these results are acceptable. This criteria is more robust than the maximisation of queues because unstable configurations are eliminated because of the time-in-the-system considerations. This result took 15 configurations as seen in table 8-e.

Ex numb	Perf	order	cheque	cash	credit	collect	changed
1	49	2	1	2	3	4	
2	61	3	1	2	3	4	1
3	66	4	1	2	3	4	1
4	63	5	1	2	3	4	1
5	67	4	2	2	3	4	2
6	65	4	3	2	3	4	2
7	51	4	2	3	3	4	3
8	65	4	2	2	4	4	4
9	68	4	2	2	2	4	4
10	67	4	2	2	2	5	5
11	69	4	2	2	2	3	5
12	69	4	2	2	2	2	5
13	66	5	2	2	2	3	1
14	67	4	3	2	2	3	2
15	57	4	2	3	2	3	3

Table 8-e : Output from WES about system investigation using default criteria.

7.4.3 Considerations about the system's limitations

The experiment described in section 7.4 was informative on a number of counts. It was concluded that WES could perform as well as MBA students given the same criterion: time in the system of entities. It was found that the use of the criterion for the maximisation of service utilisation could only work with the current default optimisation rules if there is a strict check on the

stability of the configurations selected.

Consideration of the solution process revealed that WES did not exhibit any real intelligence. The system is seen to sometimes perform "dumb" experiments. Many changes of parameters are not very sensible in the light of previous information. Also the rules deal with the global problem and do not address the individual parameter problems of the system. Many times there may be some conflict about how to handle certain bottleneck problems. Although the objective was only to propose a framework for rules to function in, it was nonetheless decided to address some of the problems more formally as described in the next section.

7.5 Towards further expertise : dealing with uncertainty

7.5.1 Issues of Indecision

The results of the ARGON experiment highlighted the problems of exhaustive search. Although WES eventually produced results that were encouraging, analysis of its solution approach are distressing. A large number of configurations were attempted and some did not follow sensibly from previous attempts if only in the methodology of stepwise approach. The current optimisation default rules in WES can only deal with one problem at a time: that of whether the last parameter change was better or worse for the system's performance. The reason for this is the intrinsic understanding by WES of the model components and how they link. If the system could gain some notion about how parameters might relate to service and possibly which service relates to given entities, then maybe the system's "intelligence" could be improved. This section 7.5 considers the use of connection statements and the manipulation of evidence factors in order to improve on the current rules in the system.

7.5.2 An approach to uncertainty

To illustrate how WES might apply more intelligent reasoning to the problem a new set of subobjectives is considered. Then a notion of connection statements is defined to provide the understanding required to address these subgoals. And finally a series of rules are suggested to resolve the conflicts that these subgoals generate. As there is no dominant method for handling uncertainty, the author has felt the liberty to implement his own heuristic methods in the form of evidence factors.

(I) Subobjectives

WES considers primarily two criteria :

- average total utilisation of services

- average total time-in-the-system of entities.

In turn, each service and each type of entity may be "suboptimised". This means that

- each service-queue can be maximised in terms of utilisation;

- each type of entity can have its time in the system minimised.

(ii) Connection statements

This requires an understanding on the part of WES that utilisation of service is connected to the number of servers, ie

given service-queue is related to a given parameter

A notion called connection of type 1 is devised :

connection(1,[Qname],[Pname],P)

where: -1 characterises the type of this connection;

- Qname is the name of the service queue considered;

- Pname is the name of the parameter that controls the number of servers at this service;

- P is an evidence factor (0-10). It measures how uniquely parameter Pname affects service Qname. For example, a score of 10 would mean that Pname measures only the number of servers for service queue Qname.

Establishing such a knowledge-base should permit WES after each experiment to consider the services that are giving problems, and relate back to the appropriate parameter.

Similarly another type of connection is considered. Some entities may use a particular service more than other entities do. So a notion of connection of type 2 is considered:

connection(2,[Ename],[Qname],P)

where this time P is an evidence measure (0-10) of the proportional usage of this service by entities of type Ename compared to others. Eg. 10 means that only entities of type Ename use this service.

Finally, the amount a parameter affects a given type of entities is devised as a connection of type 3.

connection(3,[Ename],[Pname],P)

from

connection(1,[Qname],[Pname],Pa)
connection(2,[Ename],[Qname],Pb)

where P is (Pa+Pb) // 2

These connection statements provide WES with insight into the problem structure that can be manipulated by a new set of rules.

(iii) Some rules to combine uncertain evidence

In order to address the optimisation problems, a new set of rules were devised in a file called EXPERT. EXPERT was intended to demonstrate how WES could behave more intelligently by changing more than one parameter at a time. These new rules were as follows.

Rule 1

IF utilisation of a service queue > 95%
corresponding parameter can increase value by +2
THEN assert definite move parameter value +2 with evidence +10

Rule 2

IF utilisation of a service queue > 90%
corresponding parameter can increase value by +1
THEN assert possible move parameter value +1 with evidence 10

Rule 3

IF utilisation of a service queue > 70%
corresponding parameter can increase value by +1
THEN assert possible move parameter value +1 with evidence 7

```
Rule 4
------
      IF    utilisation of a service queue  < 10%
            corresponding parameter can decrease value by -2
      THEN        assert definite move parameter value -2 with evidence -10

Rule 5
------
      IF    utilisation of a service queue  < 20%
            corresponding parameter can decrease value by -1
      THEN        assert possible move parameter value -1 with evidence -10

Rule 6
------
      IF    utilisation of a service queue  < 35%
            corresponding parameter can decrease value by -1
      THEN        assert possible move parameter value +1 with evidence -7

Rule 7
------
      IF    must look at time in the system of entities
      THEN        goto rule 11

Rule 8
------
      IF    there are definite moves
      THEN  carry out experiment with these definite changes

Rule 9
------
      IF    can find all possible moves for each parameter
            and can combine evidence P = Pa+Pb+Pc
            and P < 6 or P >-6
      THEN        assert decision

Rule 10
-------
      IF    have got this far
      THEN        must look at time in the system of entities

Rule 11
-------
      IF    time in the system  >  upper bound
            corresponding main service queue with evidence Pa
                  has utilisation > 70 %
            corresponding parameter can increase value by +1
      THEN        assert possible move parameter value +1 with evidence Pb
                  where Pb is (10+Pa) // 2
```

```
Rule 12
-------
      IF    time in the system  >  upper bound
            corresponding main service queue with evidence Pa
                  has utilisation > 60 %
            corresponding parameter can increase value by +1
      THEN        assert possible move parameter value +1 with evidence Pa

Rule 13
-------
      IF    time in the system  <  lower bound
            corresponding main service queue with evidence Pa
                  has utilisation < 50 %
            corresponding parameter can decrease value by -1
      THEN        assert possible move parameter value -1 with evidence - Pa

Rule 14
-------
      IF    there are definite moves
      THEN  carry out experiment with these definite changes

Rule 15
-------
      IF    can find all possible moves for each parameter
            and can combine evidence P = Pa+Pb+Pc
            and P < 6 or P >-6
      THEN        assert decision

Rule 16
-------
      IF    previous experiment suggested opposite change
      THEN  consider which has strongest evidence
            and assert decision
```

7.5.3 EXPERT and the ARGON problem

These EXPERT rules were used to optimise the ARGON problem. The knowledge-base presented in table 9-a was used. Appendix 11-a displays a listing of the rules used and appendix 13-f provides an illustrative consultation where it is seen how this knowledge-base is set up interactively by questioning the user. The experiment conclusion was 4 2 2 2 3 again. However, it only took three sets of experiments.

```
connection(1,[orders_qu],[order_servers],10).
connection(1,[cash_qu],[cash_servers],10).
connection(1,[cheque_qu],[cheque_servers],10).
connection(1,[credit_qu],[credit_servers],10).
connection(1,[jewel_q],[collect_servers],2).
connection(1,[collect_q],[collect_servers],8).

connection(2,[cash_customers],[cash_servers],10).
connection(2,[cheque_customers],[cheque_servers],10).
connection(2,[credit_customers],[credit_servers],10).
connection(2,[return_customers],[collect_servers],1).
connection(2,[jewel_customers],[collect_servers],2).
connection(2,[no_order_customers],[order_servers],1).

connection(3,[cash_customers],[cash_qu],10).
connection(3,[cheque_customers],[cheque_qu],10).
connection(3,[credit_customers],[credit_qu],10).
connection(3,[return_customers],[collect_q],1).
connection(3,[jewel_customers],[collect_q],2).
connection(3,[no_orders_customers],[order_q],1).
```

Table 9-a : Example connections for EXPERT

The steps involved saw WES proceed through the following stages.

Step 1: 2 1 2 3 4
- run the default configuration: 2 1 2 3 4
- results utilisations: 98% 40% 14% 1% 6% 3%

By applying rule 1 that says if utilisation is greater than 95%, make definite move to value +2.

No more considerations are investigated.

Step 2: 4 1 2 3 4

results utilisations: 66% 87% 38% 10% 17% 2%

rule 2: suggests parameter 2 increases +1 with evidence +7
rule 5: suggests parameter 4 decreases -1 with evidence -10
rule 5: suggests parameter 5 decreases -1 with evidence -10

All these are greater than |6| so they are all carried out as decisions.

Step 3: 4 2 2 2 3

results utilisations: 66% 59% 38% 35% 51% 9%

rule 5 suggest parameter 5 -1 evidence -1

As this is not enough evidence to take a decision, WES now considers further rules about the time-in-the-system of entities. The rules rely on the user inputting some extra knowledge about his expectancy of acceptable time in the system of entities. In the case of customers, it was arbitrarily decided that less than 10 minutes was very good and more than 25 would be unacceptable. The rules then proceeded to:

Step 3: further analysis.

results utilisations: 66% 59% 38% 35% 51% 9%
results time in system: 1348 1024 1219 403 321 442

Although all the view, return and jewel customers spend less than 10 minutes in the shop, no rule can apply. So the configuration is considered optimal

These results showed that uncertainty could be treated in the current framework of WES.

7.5.4 Application of EXPERT

In order to investigate the applicability of EXPERT, it was used to address the optimisation problem of a Malaysian Immigration Office. The problem was about staffing an office and very similar in nature to the ARGON case. The problem was originally chosen because the analyst of that project had spent a considerable amount of time varying every single parameter in a blind manner similar to the way WES does with its current default rules. Also, he had presented the range of all the acceptable values for each parameter for an optimal solution. In this exercise, WES did not come up with the same recommendation as the student despite being given the same criterion regarding time-in-the-system of entities. WES did however provide a solution within the range identified as acceptable. The use of default rules required 17 investigations versus the 21 of the user !! Using EXPERT, the optimal configuration was still slighlty different to the students recommendations (although within the range) but the process took only 5 steps.

7.6 Users and WES

7.6.1 Motivation

In the previous experiments, the author used WES to illustrate its capability in different situations. In the experiment described in this section, 20 non-simulationists are confronted with the ARGON problem and asked to address the same issues as before. It is investigated whether with the use of WES, they produce acceptable solutions.

Although not a pedagogical project, the experiment described in this section was also used to obtain a feeling about the system's ability to enhance the understanding and awareness to simulation issues of users that have minimal or no simulation experience.

7.6.2 Experiment Description

(i) Design and Objectives

The objective was to consider whether the use of WES by non-experts could improve their performance. Improvement is a relative concept. Therefore, the original intention had been to consider the use of a control group to assess the impact of WES. The control group would use standard tools as were available to MBA students (ie the simulation model and its own performance measures) whilst the subject group would use WES. The experiment would prove conclusive if subjects proposed significantly less wrong configurations than the control group.

The experiment was designed on a seven step format:

step 1 : Demonstration
step 2 : Knowledge Questionnaire
step 3 : Problem Description
step 4 : Initial Approach
step 5 : Experimentation
step 6 : Results
step 7 : Feedback

a. Step 1: Demonstration

The prerequisite for partaking in the experiment was no experience in discrete event simulation. In fact, only two of the control group subjects had ever used continuous simulation in a PCB manufacturing system. Subjects and control were thus presented with an introductory tutorial to simulation which quickly overviewed the concepts of simulation using the dieshop model as an example. This demonstration may be found in module T of the WES technical documentation manual.

A second tutorial followed on from this for the subject group : it described WES and how it might address simulation experimentation evaluation problems. (see Module S).

b. Step 2: Knowledge questionnaires

Volunteers answer a questionnaire that asks them :

- Why use simulation ?
- How would you use simulation ?
- What points would you need to consider carefully ?
- Where is the place of statistics in simulation ?
- Why replicate experiments ?
- List any dangers you may think of about simulation :

- What is :
  - a steady state ?
  - a run in time ?
  - a model ?
  - a run length ?
  - significance of results ?
  - utilisation of service ?
  - time in the system of entities ?

The objective was to get the subjects to focus the attention on simulation issues and gather their thoughts about the tutorial they had just received.

Step 3 : Problem Description

Volunteers are presented with the ARGON model and the same problem description as given to MBA's in their assessment. Users get a chance to use this unlinked version to understand the problem. They are not however expected to start any formal experimentation at this stage.

Step 4 : Initial Approach

Subjects and controls are asked to outline how they thought they would go about addressing the problem by describing possibly any experiment they thought appropriate and to mention points that would be important and the criteria to be used.

Step 5 : Experimentation

Controls use the model on its own and perform experimentation whilst subjects are made to use WES to approach the problem

Step 6 : Results

Volunteers are asked to indicate their recommendations to the problem with possible justifications and further considerations that may be considered. Also they are invited to make any remarks about the experiment

Step 7 : Feedback

Volunteers are recalled. They are given another (the immigration office) problem to consider. They are also asked to fill questionnaire in again. It is hoped to detect some increase in their knowledge of simulation concepts and awareness to simulation problems.

To test the design of this experiment, a pilot study was conducted as described below.

(ii) Pilot Study

Initially four candidates were chosen for a pilot study to investigate the suitability of the experiment. The objective was to consider their response to the experiment and to investigate whether any shortcomings existed in the proposed format. The four subjects were split into a control group and a subject group.

a. Control person

The control person exemplified all the speculation about the dangers of misuse of simulation in the hands of users with minimal knowledge. He declared: *" The simulation was very good and it was easy to observe the problem areas that needed changing. Once changed, the effects were easily observed. I feel this would not be possible with statistics. The simulation was very useful, quick and easy to use (csto)"*. The compliments are double-edged, as he got the wrong configuration

b. Subject Group

The subject group complained about wanting to play more with the model before using WES. They also found the demonstration tutorials too long and slow. However on the positive side, it was found that the subjects were able to handle WES to set up sensible experimentations and the results produced were also acceptable.

c. Conclusion

The problem that came apparent was the danger of individual aptitude. It would be better to measure the contribution of WES in a domain that is independent of individual influence. It was decided to make every participant his own control : ie by measuring his own improvement. The experiment would prove conclusive if with the use of WES, there was a significant reduction in the number of "wrong" configurations.

The tutorial was also to be split and implemented as program rather than as text files, as the difference produces an increase in the order of 25 times the speed.

(iii) Redesign of the experiment

The experiment was redesigned. The control group was abolished as it was desired to eliminate the personal factor. It was decided to give each user the chance to perform the role of the control group and then to introduce him to WES to see if he wanted to improve on his recommendations. The experiment now has 10 steps. (See appendices 12a & 12b).

step 1 : Demonstration
step 2 : Knowledge Questionnaire
step 3 : Problem Description
step 4 : Initial Approach
step 5 : Experimentation
step 6 : Results
step 7 : WES
step 8 : Refinement
step 9 : New Results
step 10: Feedback

Step 1 : Demonstration

At this stage only simulation concepts are introduced as was done to the control group previously.

Step 7 : WES

The tutorial on WES was now moved till after the first experimentation process. Subjects are then given a tutorial of WES for the evaluation task.

Step 8 : Refinement

Volunteers are given WES with only the evaluation task and the default criteria working in order to possibly refine the experiments they have carried out.

Step 9 : New Results

Volunteers are asked to indicate if their recommendations have changed and if so why. Again further considerations and extra remarks about the experiment are invited.

(iv) Main Study

The whole study involved 21 subjects in well over 80 hours of experimentation. Over half the subjects were professional people in engineering, accountancy or computer consultancy. The others were students in engineering or with relevant experience about working in a retail store (see table a12-1 in appendix 12-c).

7.6.3 Results

(I) Configuration

Details of the whole experiment are provided in table a12-2. Simply comparing the results of the most popular values before and after WES did not indicate anything significant about the contribution of WES. Only a sharpening of the most popular values is recorded (table a12-3). The impact of WES was better seen by the number of parameters changed. Considering that there are only 5 parameters, it appeared significant that 12 out of 17 subjects changed 3 or 4 parameters. (see table 12-a).

```
1 : **
2 : ***
3 : ****
4 : ********
```

Table 12-a : Number of parameters changed

The experiment proved conclusive when actually looking at the individual configurations suggested. It seemed significant that every subject produced a different configuration. It was also noted that 11 of these configuration are "wrong" as can be seen in table 12-b. Parameters must be within the range:

```
3 < order  < 6
1 < Cheque < 4
1 < cash   < 4
1 < credit < 5
2 < collect < 5
```

After using WES, results converged onto a number of configurations. Significantly, there was only one configuration that was considered "wrong" (see table 12-c).

From this experiment it was concluded that WES significantly helped users with minimal or no simulation knowledge to achieve results comparable to students who have some knowledge if not expertise.

+	31333 :	*
	42323 :	*
+	42325 :	*
	42333 :	*
+	43134 :	*
	43234 :	*
+	44444 :	*
	52224 :	*
+	53213 :	*
	53223 :	*
	53334 :	*
+	53336 :	*
	54324 :	*
+	62236 :	*
+	62224 :	*
+	62425 :	*
+	63225 :	*
+	74524 :	*

Table 12-b : Most popular configurations : before WES
18 different configurations
(+ denotes "wrong" configurations)

	42223 :	*******
	42234 :	*
	43223 :	***
	43224 :	*
	43323 :	*
	52223 :	**
	53223 :	**
	53224 :	*
	53233 :	**
+	53244 :	*

Table 12-c : Most popular configurations : after WES
10 different configurations
(+ denotes "wrong" configurations)
(* denotes number of times chosen)

(ii) Knowledge and Awareness Issues

The "feedback" (part 2) experiment was hoped to provide some evidence about the impact of WES on users in terms of their knowledge and awareness to simulation problems.

By comparing the subjects' initial approach to the ARGON problem, when they only had minimal exposure to simulation experimentation, with their recommendations for the IMMIGRATION OFFICE problem, it has been possible to detect an improvement in simulation knowledge (see table a12-5), and notice that awareness of simulation issues had been increased (see tables a12-4 and a12-6). They considered looking at the problem data to get a feeling of the problem and most suggested formal experimentation as necessary in order to achieve reliable results.

7.6.4 Remarks from the Subjects about the Experiment

The impression that WES contributes to making users aware of the issues in simulation may be seen through a number of comments made at the end of the experiment. These comments also reflect the usefulness of WES.

One subject actually challenged the quality of his results after the first experiment. *"It quickly becomes apparent that the longer the run, the better (more reliable) the results. Numbers in the shop appear to settle down to a steady pattern after thousands of seconds (not hundreds as I had first assumed). Running the final model in batch for longer periods suggests considerable peaks and troughs. More time is needed to run the simulation for extended periods (ie hours). (thod)"*.

The comments of this computer consultant were thus favourable to WES as he later stated: *"WES seemed an altogether better approach to the problem than using ARGON on its own. (thod)"*. This was echoed by another : *"the running of the experiment by WES is more accurate and therefore more useful than the briefer period when the operator is running the experiment and therefore of more significance" (hpal).*

The benefits of having reliable results were recognised by several: *"This experiment improved on the model by giving 'concrete' figures which showed the manner of the trade-off in a form of sensitivity analysis. Hence being very useful." (iard)*

"This gives a more definitive scenario than the previous experiment which just provided an overview of the situation. With this method it is possible to assess the amount of utilisation of

each parameter". (sdav).

"Hard statistics on long runs are of greater intrinsic value than arbitrarily stopping the simulation at certain points to 'judge' progress".(msla)

The reception of the system was favourable in terms of its teaching ability. *"Have learnt a lot. With time I would feel confident to make a judgement regarding model. Easy to use. Found the WES system particularly helpful in analysing how well I had done". (msul).* Awareness of what is involved was indicated in the comment : *"WES System does help considerably in having a reasonable idea as to what is needed to do a simulation and how long to do one". (stay).*

"Feedback from experiment gave a clearer interpretation of the experiments undertaken. The need for a large number of experiments is made obvious".(nver)

This was reflected as a general lesson which appeared to be learnt by many : *"It is difficult to come up with the optimal solution without a large amount of experimentation".(fatm)*

It cannot be said that everybody took the lesson home. One particular subject thought the deficiencies of his result could be overcome by a different visual lay out : *"Visual side of 1st experiment maybe include a different form to show how many of your staff are actually being used."(cmit).*

Overall the system was described as *"very useful to the layman"* (mhas).

7.6.5 Conclusion

Users with little or no simulation knowledge have been able to make use of the demonstration and explanation facilities in order to use the system to achieve a level of studies comparable to students in simulation courses. There is some modest evidence that WES increases knowledge but it certainly makes people more aware of dangers of simulation. Overall the system appeared usable

7.7 Conclusion

In this chapter, a number of experiments were conducted in order to derive some appreciation about the current use of the system and the potential of this approach of adding expertise to simulations.

In the first series of experiments, a number of retro-active studies were performed in order to assess the suitability of WES. It was discovered how WES can overcome a number of problems relating to :

- bad data transcriptions;

- bad design;

- steady state issues (lack of consideration ,overkill);

- perfect memory.

In the second phase, WES was applied to a data-driven simulation system. It was found that WES was portable to other packages and that combined to data-driven simulations, the notions of automatic learning and understanding was a very real possibility. WES could automatically learn about a system, evaluate it and present it to a user who need know nothing about this model.

In the third stage, the ability of WES to perform as well as MBA students in the optimisation of a model was considered. It was discovered that it was possible for the WES system to perform as well.

As a fourth consideration, the use of more sophisticated rules that could deal with uncertainty were illustrated to show the potential of WES, given enough time and effort. It was seen how the performance could be made more intelligent.

Finally as an attempt to examine WES potential as a pedagogical tool, an experiment was devised to see whether WES could assist non-knowledgeable simulation users to perform as well as students who have attended simulation courses. The experiment proved conclusive and a improvement in awareness of simulation issues could be detected.

It is concluded that WES already has current usefulness and offers great potential with its application to other simulation packages and the development of more sophisticated rules that consider uncertainty.

End of Chapter

Chapter 8

Conclusions and Future Work

8.1 Introduction

This chapter presents a summary of the research carried out for this thesis, together with conclusions and recommendations for future work. The objective is to try and establish the significance of what has been achieved.

8.2 Research Summary

Simulation studies cycle through the phases of formulation, programming, verification and validation, experimental design and analysis, and implementation. This thesis has been concerned with developing methods to enhance the practice and support for the experimental design and analysis phase of a study. The investigation focussed on the introduction of AI techniques to this phase, where previously there existed little support. The reason for this approach was the realisation that the experimentation process in a simulation study can be broken down into a reasoning component and a control of execution component. In most studies, a human user would perform both of these (though some existing systems already implement a control of execution component). The involvement of a reasoning process attracted the notion of artificial intelligence or at least the prospective use of its techniques.

The first stage of the research was to undertake a detailed survey of the literature on the topics relevant to this work. This covered artificial intelligence and its techniques, decision support, experimentation, simulation and its relationship to these other topics. The background work highlighted the potential of AI techniques in simulation and encouraged further investigation into their use in the context of experimental design and analysis.

The second stage of the research was to consider the development of a support system for

experimental design and analysis that had human intelligence and machine control of execution. The consequence of this work was a semi-structured decision making environment in the form of a controller that requested human input. The controller was made intelligent when it was linked to a non-procedural (PROLOG) program that provided remote intelligent input from either the user or default heuristics. The intelligent controller was found to enhance simulation experimentation because it ensures that all the steps in the experimental design and analysis phase take place and receive appropriate input.

The third phase was to adopt the view that simulation experimental design and analysis may be enhanced through a system that had machine intelligence and expected human control of execution. This provided the framework of an advisor that adopted a consultation expert system paradigm. Users were advised on how to perform simulation experimentation. Default reasoning strategies were implemented to provide the system with advisory capabilities in the tasks of prediction, evaluation, comparison, sensitivity analysis, transient behaviour, functional relations, optimisation.

The fourth step unified the controller and the advisor to present an integrated system with both machine intelligence and machine control of execution. User involvement in the experimentation process was reduced considerably as support was provided in both the reasoning and control of execution aspects. Additionally, this integrated system supports facilities for refinement that aim at turning the system's knowledge into expertise. It became theoretically possible for other simulation experts to teach the system or experiment with their own rules and knowledge.

The fifth stage considered making the knowledge of the system available to the user, thereby turning the system into a teacher and providing pedagogical support. Teaching was introduced through explanation and demonstration. The explanation facility used a mixed approach : it combined a first time response explanation facility to "how" and "why" questions with a menu driven information system facility for "explain" requests or further queries. The demonstration facility offers tutorials on the use of the system and how to carry out an investigation of any of the tasks that the system can address.

The sixth and final part of the research was to collect some empirical results about the performance of the system. The objective was to obtain some appreciation of how much support is actually provided to the user. Some experiments were performed retroactively on existing studies. The system was also linked to a data-driven simulation package that permitted evaluation using some large scale industrial applications. The system's performance was measured by its ability to perform as well as students with simulation knowledge but not necessarily expertise. The system was also found to assist the user with little or no simulation knowledge to perform as well as students with knowledge.

8.3 Conclusions

A number of conclusions can be drawn from the work presented in this thesis. These conclusions come under several headings.

8.3.1 Enhancing Experimentation with an Intelligent Controller

By adopting an environment where there was remote input from one micro computer into another that had process control, it has been possible to illustrate the mechanisms of the experimental design and analysis process as requiring an intelligent reasoning component and a control of execution component. The essence of the controller is to enforce a formal and procedural framework on the control of execution of experiments. The experimentation process is made to go through a number of steps every time thus ensuring that proper practice takes place. Some simulation packages already implement such formalism. However, this controller can be made intelligent by enhancing the quality of the input. By using a remote (non-procedural) PROLOG program to obtain input, it is possible to make use of the non-procedural characteristics of AI languages to implement default mechanisms when presented with input from a user with little or no expertise. PROLOG is well suited for this kind of programming. It is concluded that the introduction of a formal controller can enhance the practice of simulation experimental design and analysis.

8.3.2 Advisory Systems on an AI Paradigm

It has been possible to show that a consulation framework can encompass the steps that are used in the practice of simulation experimentation. By analysing experts, it has been possible to derive models of intelligent reasoning behaviour. The notions of strategies and the process of strategic reasoning appear to provide a viable understanding of the problem solving activities involved in the design of experiments and their analysis. It has been possible to make use of expert system principles of modelling and problem solving methodologies to incorporate this knowledge and understanding. Although the knowledge of the system is currently limited to

heuristics, the implementation of default rules has illustrated that a number of tasks may be supported to a level of expertise comparable to students who have attended simulation courses. It is concluded that the expert system framework provides an encouraging avenue to explore in the support of experimental design and analysis.

8.3.3 Integrated Experimenter

It has been shown that it is possible to significantly reduce the amount of human involvement required in the experimentation process. The use of default values, rules and strategies has made it possible for a user with little or no simulation knowledge to perform experiments to a level comparable of knowledgeable but non-expert students. A general linking procedure has been devised that would allow, in theory at least, any high-level language written simulation model to link into the system. It was also possible to increase the system understanding of the model environment by linking the system to a data-driven simulation package and provide the system with automatic learning capabilities. It is concluded that the development of a system that links the concepts of a "controller" and "advisor" provides a powerful user-friendly environment for the execution of the experimental design and analysis phase of a simulation study.

8.3.4 Transfer of Expertise

Availability and explicity of knowledge can be argued as key factors in the quality of decision making. By developing a mechanism that allows a human expert to transfer knowledge to the system, it is concluded that the system can tend towards expertise.

By combining an explanation facility with the use of a menu-driven information system facility at secondary explanation level, it has been found that support knowledge may be accessed at all times. It is concluded that user understanding may be assisted by making machine knowledge available to the user.

8.4 Future Research

In this final section, those areas thought to be worthy of future research are outlined. These areas come under two major headings : intuitive strategies and understanding of model environment.

8.4.1 Intuitive Strategies

The performance of the system will be reflected by the amount of procedural and non-procedural knowledge that the system can deal with. At the controller level, alternative frames can be input. At the advisory level, many more production rules can be input. Experts may be sampled from a variety of backgrounds in order to enhance the systems ability. However, it is generally recognised that experts often follow hunches, therefore intuitive strategies is an area that would provide interesting work. It is likely that the use of probabilities will be necessary.

8.4.2 Understanding the Model Environment

At present, the system assumes the existence of a verified and validated model. It is conceivable to think of the use of the system in the development stage. In parallel with this thesis, work has been undertaken at Warwick University in another doctoral program in the context of automated model building. Testing and evaluation of intermediate models could be conceived.

Development of the work illustrated by the linking exercise to the data-driven models would constitute an interesting extension as notions of simulation expertise at all stages of the development cycle of simulation studies can be considered. This would provide the system with a complete knowledge and understanding of the model and its environment. The quality of recommendations would thus be increased. Such work would provide important foundations for the concept of an expert "system" simulation analyst.

This thesis has demonstrated a robust framework for an expert advisory system for experimental design of a simulation. It is suggested that the approach adopted in this thesis is applicable to other areas of OR expertise. By modelling the decision-making processes involved in the

application of different OR techniques, support expertise may be provided at the appropriate decision points by the development of advisory systems.

THE END

Bibliography

Adelsberger, H.H., 1984, "Prolog as a Simulation Language", in *Proceedings of the 1984 Winter Simulation Conference*, S. Sheppard, M. Poach, D. Pegden (Eds), 501-504.

Adelsberger, H.H., 1986a, "Rule based object oriented simulation systems", in *Simulation Series*, Vol 17, NO.1 107-112, Luker and Adelsberger (editors).

Adelsberger, H.H., 1986b, "Introduction to Artificial Intelligence", in *Simulation Series*, Vol 17, No.1 141-143, Luker and Adelsberger (Editors).

Adkins, G.W., 1982, "Variance reduction in estimating the mean flow time in open queuing network simulations", in the *Proceedings of the 1982 Winter Simulation Conference*, pp 451-454, Highland, Chao and Madrigal (editors), I.E.E.E.

d'Agapeyeff, A., 1984, *"Report to the ALVEY directorate on a short survey of Expert systems in UK Business"*, Supplement to Vol. 4 ALVEY.

Ahmad, A. and R.D. Hurrion, 1988, "Automatic model generation using a Prolog model-base", in the *Proceedings of the SCS Multiconference on Artificial Intelligence and Simulation : The Diversity of Applications*, (T. Henson, Ed.),137-142, The Society for Computer Simulation International, San Diego, Calif.

Allen, R.W., 1985, "A frequency domain program for analysing simulation models", in *Modelling and Simulation on Microcomputers*, pp 3-8, R.G. Lavery (editor).

Alter, S.L., 1980, *Decision Support Systems - Current Practice and Continuing Challenges*, Addison Wesley.

Amer, P.D., 1982, "Providing statistical confidence in the evaluation of alternatives", in *Simulation*, Vol. 38, No.1, 13-18.

Balci, O and R.E. Nance, 1987, "Simulation model development environments : a research prototype", in *Journal of Operational Research*, Vol. 38, No.8, 753-764.

Balmer D.W. and R.J. Paul, 1986, "Casm - the right environment for for simulation", *Journal of the Operational Research Society*, Vol 37, 443-452.

Barr, A. and E.A. Feigenbaum, 1982, *The Handbook of Artificial Intelligence*, Pitman Books Ltd.

Barstow, D.R. 1982, "The roles of knowledge and deduction in algorithm design", in *Machine intelligence 10*, Chapter 18 361-381, Hayes, Michie and Pao (editors), Ellis Howood

Beal, C.W., 1982, "A regression technique for determining steady state conditions in time series simulations", in the *Proceedings of the 1982 Winter Simulation Conference*, pp 439-448, Highland, Chao and Madrigal (editors), I.E.E.E.

Bell, M.Z., 1985, "Why expert systems fail", in *Journal of Operational Research*, Vol. 36, No. 7, 613-619.

Bell, P.C., 1985, "Visual Interactive Modelling in Operational Research Successes and Opportunities", in *Journal of Operational Research*, Vol. 36, No.11, 975-982.

Benet, J.L., 1983, *Building Decision Support Systems*, Addison -Wesley.

Birnie, W., 1987, "Consulting the expert system", in *Civil Engineering*, July, p22-23.

Birtwistle, G.M., 1979, *Discrete Event Modelling on Simula*, The MacMillan Press Ltd, London.

Birtwistle, G. and J. Kendall, 1986, "A view of Lisp", in *Simulation Series*, Vol 17, No.1, 157-162, Luker and Adelsberger (editors).

Blakemore, J.W., S.B. Dolins and P.R. Thrift, 1986, "A general purpose robotics vehicle simulator", in *Simulation Series*, Vol. 18, No.1, 151-161, Kerckhoffs, Vansteenkiste and Zeigler (Editors).

Blightman, R., 1987, "Where now with simulation", in *Journal of Operational Research*, Vol. 38, No.8, 769-770.

Bobillier, P.A., B.C. Kahan, A.R. Probst, 1976, *Simulation with GPSS and GPSS V*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Bond, A., (editor), 1981, *Machine Intelligence - State of the Art Report*,, Pergamon Infotech.

Bondi, A.B. and A.A.B. Pritsker, 1986, "Alternate analyses of dynamic models", in *Simulation Series, "AI applied to Simulation"*, Vol. 18, No.1 32-41, Kerckhoffs, Vansteenkiste and Zeigler (editors).

Bos, A.M., 1986, "Formula manipulation in the simulation of bond graph models of mechanical systems", in *Simulation Series, "AI applied to Simulation"*, Vol. 18, No.1 27-31, Kerckhoffs, Vansteenkiste and Zeigler (editors).

Bowen, H.C., R.J. Fenton, M.A.M. Rogers, R.D. Hurrion and R.J.R. Secker, 1979, "Interactive Computing as an Aid to Decision Makers", in *OR '78*, pp 829-842, editors K.B. Harley, North-Holland Publishing Company.

Brachman, R.J., 1979, "On the epistemelogical status of semantic networks", in *Associative Networks* N Findler (editor), Academic Press.

Brookings, A.G., 1984, "The fifth generation game", in *Expert Systems* by R Forsyth: Chapman and Hall (Publishers).

Brown J.C., 1978, "Visual interactive simulation: further developments towards a generalised system and its use in 3 problem areas associated with a high technology company, *MSc Thesis*, SIBS, Warwick University.

Buchanan, B.G., 1982, "New research on expert systems", in *Machine Intelligence 10*, Chapter 14, 269-299, Hayes, Michie and Pao (editors).

Buchanan, B.G., D.Barstow, R. Bechtal, J. Bennet, W. Clancey, C. Kulihowski, T. Mitchell and D. Waterman, 1983, "Consulting an Expert System", in *"Building Expert Systems"*, by Hayes, Roth et al, Chapter 5 127-167.

Bulgren, W.G., 1982, *Discrete System Simulation*, Prentice-Hall, Inc., Englewood Cliffs, N.J.

Campbell, R.A., 1986, "Development of an expert system for simulation model selection", in *Simulation Series, "Intelligent Simulation Environments"*, Vol. 17, No.1, 121-122, Luker and Adelsberger (editors).

Clark, K.L. and J.G. McCabe, 1982, "Prolog : a language for implementing expert systems", in *Machine Intelligence 10*, Chapter 23, 455-470, Hayes, Michie and Pao (editors).

Cleary, J., K.S. Goh and B. Unger, 1985, "Discrete event simulation in Prolog" in *A.I., Graphics and Simulation*, 8-13, G. Birtwistle (editor).

Clementson, A.T., 1974, *Computer Aided Programming for Simulation*, The Lucas Institute for Engineering Production, Birmingham University.

Clementson, A.T., 1980, *Extended Control and Simulation Language*, The Lucas Institute for Engineering Production, Birmingham University.

Clocksin, W.F. and C.S. Mellish, 1984, *"Programming in Prolog"*, Springer-Veilag, Berlin Heidelberg, New York, Tokio.

Cloutier, T, M. Bourgarelt, C. Roy, E. Cerny, 1987, "Performance vs flexibility in a hierarchical multi-level VLSI simulator", in *Simulation Series, "Simulation and AI"*, Vol. 18, No.3, 92-97, Luker and Birtwistle (editors).

Cohen, P.R. and M.R. Grinberg, 1983, "A Theory of Heuristic Reasoning about Uncertainty", in *The A.I. Magazine* Summer, pp17-24.

Colmerauer, A., H. Kanoui, M. van Caneghem, 1973, "Prolog theoretical principles and current trends", *Technology of Science of Informations*, Vol. 2, No 4, 256-292.

Cole, M., 1986, "A first look at an expert computer system", *Accountancy*, July, pp129-131.

Conway, R., 1963, "Some tactical problems in digital simulation", *Management Science*, Vol. 10, pp 47-61.

Conway, R., B. Johnson and W. Maxwell, 1959, "Some problems of digital systems simulation", in *Management Science*, Vol. 6, pp 92-110.

Cox, P.R., 1984, "How we built Micro Expert", in *Expert Systems*, by Richard Forsyth, Chapter 8, Chapman and Hall, London.

Crookes, J.G., 1982, "Simulation in 1981", *European Journal of Operational Research*, Vol. 9, 1-7.

Crookes, J.G., 1987, "Generators, generic models and methodology", *Journal of Operational Research*, Vol. 38, No. 8, 765-768.

Crookes, J.G., D.W. Balmer, S.T. Chew and R.J. Paul, 1986, "A three-phase simulation system written in Pascal", in *Journal of Operational Research*, Vol. 37, No. 6, 603-618.

Curran, S. and R. Curnow, 1983, *The Penguin Computing Book*, Penguin Books.

Davey, P.G., 1982, "Robot R&D in the U.K.", *SERC report*, February.

Davies, H. and R. Davies, 1987, "A simulation model for planning services for renel patients in Europe", in *Journal of Operational Research*, Vol. 38, No. 8, 701-712.

Davis, E.L., 1986, "Fifth Generation", in *Simulation Series, "Intelligent Simulation Environment"*, Vol. 17, No. 1, 113-117, Luker and Adelsberger (editors).

Davis, K.R. and P.G. McKeown, 1984, *Quantitative Models for Management*, Kent Publishing Company, Boston.

Davis, R., 1977, "Applications of meta level knowledge to the construction, maintenance and use of large knowledge bases", *Doctoral Thesis*, Chapters 4,5 & 6, pp69-195.

Davis, R., 1980a, "Meta-Rules : Reasoning about Control", in *Artificial Intelligence*, Vol. 15, 179-222.

Davis, R., 1980b, "Meta-Rules : Reasoning about Rules", in *Artificial Intelligence*, Vol. 15, 223-239.

Davis, R., B. Buchanan and E. Shortliffe, 1977, "Production Rules as a Representation for a Knowledge -Based Consultation Program", in *Artificial Intelligence*, Vol. 8, 15-45.

Davis, R. and D.B. Lenat, 1982, "Explanation", in *Knowledge Based Systems for Artificial Intelligence*, R. Davis and D.B. Lenat (Eds), McGraw-Hill, New-York, 259-286.

Deutsch, T., Futo, I. and I. Papp, 1986, "The use of TC-Prolog for medical simulation", in *Simulation Series, "Intelligent Simulation Environments"*, Vol. 17, No.1, 29-34, Luker and Adelsberger (editors).

Doman, A., 1988, "OBJECT-PROLOG: Dynamic object-oriented representation of knowledge", in the *Proceedings of the SCS Multiconference on Artificial Intelligence and Simulation : The Diversity of Applications*, (T. Henson, Ed.), 83-88, The Society for Computer Simulation International, San Diego, Calif.

Doukidis, G.I., 1987, "An anthology on the homology of simulation with artificial intelligence", in *Journal of Operational Research Society*, Vol. 38, No. 8, 701-712.

Doukidis, G.I. and R.J. Paul, 1985, "Research into Expert Systems to Aid Simulation Model Formulation", in *Journal of Operational Research Society*, Vol. 36, No.4, 319-325

Doukidis G.I. and R.J. Paul, 1986, "Experiences in automating the formulation of discrete event simulation models", in *Simulation Series, "AI applied to Simulation"*, Vol. 18, No.1, 79-90, Kerckhoffs, Vansteenkiste, Zeigler (editors).

Doyle, J., 1979, "A truth maintenance system", in *Artificial Intelligence*, Vol. 12, 231-272.

Doyle, J., 1983, "Methodological Simplicity in Expert Systems Construction : the case of judgements and reasoned assumptions", in *The A.I. magazine*, Summer, pp39-43.

Dudewicz, E.J., 1977, "New procedures for selection among (simulated) alternatives", from the proceedings of the *Winter Simulation Conference*, pp 58-62.

Eden, C., S. Jones and D. Sims, 1983, *Messing about in Problems*, Pergamon Press.

Eden, C., H. Williams and T. Smithin, 1986, "Synthetic wisdom: the design of a mixed mode modelling system for organisational decision making", in *Journal of Operational Research Society*, Vol. 37, No.3, 233-241.

Elmaghraby, A.S. and V. Jagannathan, 1985, "An expert system for simulationists", in *A.I., Graphics and Simulation*, pp 106-109, G. Birtwistle (editor).

Erickson Jr, S.A., 1986, "Fusing AI and Simulation in military modeling", in *Simulation Series, "AI applied to Simulation"*, Vol. 18, No.1, 140-150, Kerckhoffs, Vansteenkiste and Zeigler (editors).

Feder, B., 1986, "Artificial Intelligence is on Wall Street's Mind", in *The Globe and Mail*, Thursday 28 August, p.136.

Feigenbaum, E., 1971, "On generality and problem solving", in *Machine Intelligence*, No. 6. Feigenbaum, E, B.G. Buchanan and J. Lederberg, 1971, "Generality and Problem Solving: A Case Study Using the DENDRAL Program", in *Machine Intelligence 6*, 165-190.

Feigenbaum, E. and P McCorduck, *The Fifth Generation*, Addison Wesley, Reading, MA.

Feller W. 1950, *An introduction to probability theory and its applications* Third edition, volume 1, John Wiley and Sons, Chichester.

Fishman, G.S., 1973, *Concepts and methods in discrete event digital simulation*, John Wiley & Sons, Inc., Canada.

Fishman, G.S., 1976, "Statistical Analysis of Multiserver Queueing Simulations", in *O.R. Quarterly*, Vol. 27, No. 4, 1005-1013.

Fjellheim, R.A., 1986, "A knowledge based interface to process simulation", in *Simulation Series, "AI applied to Simulation"*, Vol. 18, No. 1, 97-102, Kerckhoffs, Vansteenkiste and Zeigler (editors).

Flitman, A., 1983, "A guide to Unix Prolog", *Third year BSc computer science project*, University of York, England.

Flitman, A., 1984-86, "Communications via Asynchronous Adapter", in *Working papers on Fortran-Prolog Link*, PhD work, SIBS, University of Warwick, England.

Flitman A.M. 1986, "Towards the Application of Artificial Intelligence Techniques for Discrete Event Simulation", *PhD Thesis*, University of Warwick, England.

Flitman A.M. and R.D. Hurrion, 1987, "Linking discrete-event simulation models with expert systems", *Journal of the Operational Research Society*, Vol 38, 723-732.

Foster, W., 1986, "Expert systems for industrial applications - part 2", in *Simulation*, Vol. 46, No. 1, 27-30.

Forsyth, R., 1984, *Expert Systems : Principles and case studies*", Chapman and Hall, London.

Frank, M.V., 1986, "Application of artificial intelligence to improve plant availability" in *Simulation Series*, Vol. 17, No.1, 92-97, Luker and Adelsberger (editors).

Friedman, L.W., 1984a, "Multivariate simulation output analysis: past, present and future", from the proceedings of the *Winter Simulation Conference*, pp 277-281, S.Sheppard, U. Pooch, D. Pegden (editors).

Friedman, L.W., 1984b, "Establishing functional relationships in multiple response simulation: the multivariate general linear metamodel", from the proceedings of the *Winter Simulation Conference*, pp 285-289, S.Sheppard, U. Pooch, D. Pegden (editors).

Fuchi, K., 1983, "Outline of research and development plans for fifth generation computer systems", (English Translation) in *BYTE Publications Inc.*, November, pp396-401.

Fujiwara, R. and T. Sakaguchi, 1986, "An expert system for power system planning", in *Simulation Series*, Vol. 18, No.1, 174-177, Kerckhoff, Vansteenkiste and Zeigler (editors).

Futo, I., I. Papp and T. Szeredi, 1986, "The microcomputer version of TC-Prolog", in *Simulation Series, "Intelligent Simulation Environments"*, Vol. 17, No. 1, 123-128, Luker and Adelsberger (editors).

Futo, I., T. Gergely and T. Deutsh, 1986, "Logic modelling", in, *Simulation Series, "AI applied to Simulation"*, Vol. 18, No. 1, 114-129, Kerckhoffs, Vansteenkiste and Zeigler (editors).

Futo, I., 1988, "ALLEX, an Expert System Shell to support Simulation", in the *Proceedings of the European Simulation Multiconference, "Simulation Environments and Symbol and Number Processing on Multi and Array Processors"*, 173-178, Huntsinger, Karplus, Kerckhoffs and Vansteenkiste (editors), Simulation Council Inc.

Gafarian, A.V., C.J. Ancker and T. Morisaku, 1977, *The problem of the initial transient with respect to mean value in digital computer simulation and the evaluation of some proposed solutions*, Technical report no. 77-1, University of Southern California.

Gaines, B.R., "Expert systems and simulation in industrial applications", in *Simulation Series*, Vol. 17, No. 1, 144-149, Luker and Adelsberger (editors).

Gaines, B.R. and M.L.G. Shaw, 1985, "Expert systems and simulation", in *A.I., Graphics and Simulation*, pp 95-101, G. Birtwistle (editor).

Gambling, T., 1985, "Expert systems : Stone age rules OK? ", in *Accountancy*, July, pp125-127.

Gardenier, T.K., 1982, "Some uses of statistics in simulation", in *Simulation Series, Computer Modelling and Simulation Principles of Good Practice*, Vol. 10, No. 3, 129-140, by T.Mc Leod.

Gaschnig, J., 1982, "Application of the PROSPECTOR system to geological exploration problems," in *Machine Intelligence 10*, Chapter 15, 301-323, Hayes, Michie and Pao (editors).

Georgeff, M.P., 1983, "Strategies in heuristic search", in *Artificial Intelligence* Vol. 20, pp 393-425.

Gevarter, W.B., 1982, "An overview of Expert Systems" *National Bureau of Standards*, Washington.

Glicksman, J., 1986, "A simulator environment for an autonomous land vehicle", in *Simulation Series, "Intelligent Simulation Environments"*, Vol. 17, No.1, 53-57, Luker and Adelsberger (editors).

Glynn, P.W., 1982, "Coverage error for confidence intervals arising in simulation output analysis", in the *Proceedings of the 1982 Winter Simulation Conference*, pp 369-375, Highland, Chao and Madrigal (editors), I.E.E.E.

Goldberg, A. and D. Robson, 1983, *Smalltalk - 80, The Language and its Implementation*, Addison Wesley.

Goodman, J., 1988, "Simulation of minimum sample sizes for sampling procedures based on the maximum likelihood method", in *Modelling and Simulation on Microcomputers*, pp 50-51, R.G. Lavery (editor).

Gordon, G., 1978, *System Simulation*, Second edition, Prentice Hall, New Jersey.

Grant, T.J., 1986, "Lessons for O.R. from A.I. - A scheduling case study", in *Journal of Operational Research*, Vol. 37, No.1, 41-57.

Grassman, W.K., 1982, "Initial bias and estimation error in discrete event simulation", in the *Proceedings of the 1982 Winter Simulation Conference*, pp 377-384, Highland, Chao and Madrigal (editors), I.E.E.E.

Gray, P. and T. Borovits, 1986, "The contrasting roles of Monte Carlo simulation in decision support systems", in *Simulation*, Vol. 47, No. 6, 233-239.

Green, J.O., 1984, "Making computers smarter", in *Popular Computing*, January, pp 97-104.

Greenlaw, Herron and Rawdon, 1962, *Business Simulation*, Prentice Hall.

Groen A., H.J. van den Herik, A.G. Hofland, E.J.H. Kerckhoffs, J.C. Stoop and P.R. Varkevisser, 1986, "The integration of simulation and knowledge-based systems", in *Simulation Series*, Vol. 18, No.1, 189-197, Kerckhoffs, Vansteenkiste and Zeigler (editors).

Gupta, S.S. and J.C. Hsu, 1984, "A computer package for ranking, selection, and multiple comparisons with the best", from the proceedings of the *Winter Simulation Conference*, pp

251-257, S.Sheppard, U. Pooch, D. Pegden (editors).

Haddock J., 1987, "An expert system framework based on a simulation generator", *Simulation*, Vol 48, 45-53.

Haessler, R.W., 1983, "Developing an industrial grade heuristic problem solving procedure", in *Interfaces*, Vol. 13, No. 3, 62-71.

Hall, R., 1978, "Simple techniques for constructing explanatory models of complex systems for policy analysis", in *Dynamica*, Vol. 4, Part 3.

Hammond, P., 1982, "Appendix to PROLOG: a language for implementing expert systems" in *Machine Intelligence 10*, 471-475, Ellis Horwood.

Hardman, L., 1985, "Expert User-Friendly", *Systems International*, April.

Harrison, N., 1984, "Knowledge base builders", in *Systems International*, August, pp 56-59.

Hayes-Roth,F.; D. Waterman ; D. Lenat (Eds), 1983, *Building Expert Systems*, Addison-Wesley, Reading MA.

Hebditch, D., 1984, "How friendly are expert systems", in *Datamation*, 198/15-198/19.

Heikes, R.G., D.C. Montgomery and R.L. Rardin, 1976, "Using common random number in simulation experiments - an approach to statistical analysis", in *Simulation*, Vol. 27, No 3, 81-85.

Henrickson, J.O., 1983, "The Integrated Simulation Environment", *Operations Research*, Vol. 31, No 6, 1053-1073.

Henson, T. (editor), 1988, in the *Proceedings of the SCS Multiconference on Artificial Intelligence and Simulation : The Diversity of Applications*, (T. Henson, Ed.),137-142, The Society for Computer Simulation International, San Diego, Calif.

Hill, T.R., and S.D. Roberts, 1987, "A prototype knowledge-based simulation support system", in *Simulation*, Vol. 48, No.4, 152-161.

Hollocks, B., 1983, "Simulation and the micro", in *Journal of the Operational Research Society*, Vol. 34, No. 4, 331-343.

Hooper, J.W., 1986a, "Strategy related characteristics of discrete languages and models", in *Simulation*, Vol. 46, No.4, 153-159.

Hooper, J.W., 1986b, "Activity scanning and the three phase approach", in *Simulation*, Vol. 47, No.5, 210-211.

Huntsinger, R.C., W.J. Karplus, E.J. Kerckhoffs and G.C. Vansteenkiste, (editors) 1988, "Simulation Environments and Symbol and Number Processing on Multi and Array Processors", *Proceedings of the European Simulation Multiconference*. Simulation Council Inc.

Hurrion, R.D., 1976, The design, use and required facilities of an interactive visual computer simulation language to explore production planning problems, *PhD Thesis*, University of London.

Hurrion, R.D., 1980, "An interactive visual simulation system for industrial management", in *European Journal of Operational Research*, No. 5, pp 89-93.

Hurrion, R.D., 1981, "Visual interactive simulation using a microcomputer", in *Computing and Operations Research*, Vol. 8, No.4, 267-273.

Hurrion, R.D. and R.J. Secker, 1978, "Visual Interactive Simulation : An Aid to Decision Making," *Omega*, Vol 6, 419-426.

Hurrion, R.D. and C. Moreira de Silva, 1980, "*A bird's eye view of the planning functions at Huddersfield works (organic division) - some research prospects*", S.I.B.S., University of Warwick.

Iman, R.L. 1982, "Considerations with regard to input variables for computer simulations", in the *Proceedings of the 1982 Winter Simulation Conference*, pp 597, Highland, Chao and Madrigal (editors), I.E.E.E.

Institute of New Generation Computer Technology, 1983, *Outline of research and development for 5th generation computer systems*, INGCT.

Istel, 1987, *WITNESS User manual*, Istel Ltd.

Johnston, J., 1983, "Can they do it", in *Datamation*, July, 161-170.

Johnston, R., 1985, "How close are we to thinking machines? ", in *Computer Talk*, November, pp 16-17.

Jones, P.L.K., 1984, "REVEAL: An expert system support environment", in *Expert Systems*, by Richard Forsyth; Chapman and Hall, London.

Keen, P.G.W. and M.S. Scott Morton, 1978, *Decision Support Systems - An organisational perspective*, Addison-Wesley.

Kelton, W.D., 1984, "Steady-state confidence interval methodology: a forum on theory, practice, and prospects", from the proceedings of the *Winter Simulation Conference*, pp 243-250, S.Sheppard, U. Pooch, D. Pegden (editors).

Kelton, W.D., 1985, "Analysis of Simulation Output Data", from the proceedings of the *Winter Simulation Conference*, pp 33-35, Gantz, Blais, Soloman (editors).

Kerckhoffs, E.J.H., G.C. Vansteenkiste and B.P. Zeigler, (editors), 1986, The Proceedings of the European Conference in *Simulation Series, "AI applied to simulation"*, Vol. 18, No. 1.

Kerckhoffs, E.J.H., and G.C. Vansteenkiste, 1986, "The impact of advanced information processing on simulation - an illustrative review", in *Simulation*, Vol. 46, No.1, 17-26.

Kerckhoffs, E.J.H., G.C. Vansteenkiste and B.P. Zeigler, 1986, "General considerations on AI applied to simulation", in *Simulation Series, "AI applied to Simulation"*, Vol. 18, No.1, ix-xii, Kerckhoffs, Vansteenkiste and Zeigler (editors).

Kettenis, D., 1988, "The COSMOS Simulation Environment", In the *Proceedings of the European Simulation Multiconference, "Simulation Environments and Symbol and Number Processing on Multi and Array Processors"*, 394-399, Huntsinger, Karplus, Kerckhoffs and Vansteenkiste (editors), Simulation Council Inc.

Khoshnevis, B. and W.M. Austin, 1987, "An intelligent interface for systems dynamics modelling", in *Simulation Series*, Vol. 18, No.3, 81-86, Luker and Birtwistle (editors).

Khoshnevis, B. and A.P. Chen, 1986, "An expert simulation model builder", in *Simulation Series*, Vol. 17, No.1, 129-132, Luker and Adelsberger (editors).

Khoshnevis, B. and A.P. Chen, 1987, "An automated simulation modelling system based on AI techniques", in *Simulation Series, "Simulation and AI"*, Vol. 18, No.3, 87-91, Luker and Birtwistle (editors).

Klahr, P., 1986, "Expressibility in Ross: An object-oriented simulation system", in *Simulation Series, "AI applied to Simulation"*, Vol. 18, No.1, 136-139, Kerckhoffs, Vansteenkiste and Zeigler (editors).

Kleijnen, J.P.C., 1977, "Design and analysis of simulations: practical statistical techniques", in *Simulation*, Vol. 28, No.3, 81-90.

Kleijnen, J.P.C., 1975, *Statistical Techniques in Simulation, Part II*, Marcel Dekker.

Kleijnen, J.P.C., 1981, "Regression analysis for simulation practitioners", in *Journal of Operational Society*, Vol. 32, 35-43.

Kleine, H. 1977, "A vehicle for developing standards for simulation programming", from the proceedings of the *Winter Simulation Conference*, pp 730-741.

Kleinmuntz, D.N. and B. Kleinmuntz, 1981, "System simulation, decision strategies in simulated environments", in *Behavioral Science*, Vol. 26, pp 294-304.

Kleissner, K. and M. Stumpter, 1986, "SCCA: Simulation environment for concurrency control algorithms in distributed databases", in *Simulation Series, "Intelligent Simulation Environments"*, Vol. 17, No.1, 47-52, Luker and Adelsberger (editors).

Kornecki, A., 1988, "Prolog Based Air Traffic Control Expert/Simulator", in the *Proceedings of the European Simulation Multiconference, "Simulation Environments and Symbol and Number Processing on Multi and Array Processors"*, 217-236, Huntsinger, Karplus, Kerckhoffs and Vansteenkiste (editors), Simulation Council Inc.

Kornell, J., 1987, "Reflections on using knowledge based systems for military simulation", in *Simulation* Vol. 48, No.4, 144-148.

Kowalski, R.A., 1974, "Predicate logic as a programming language", in the *Proceedings of the IFIP 77 conference*, 569-574, North Holland.

Kowalski, R.A., 1979 *Logic for Problem Solving*, North Holland.

Kunz, J.C., R.J. Fallat, D.H. Mc Clung, J.J. Osborn, B.A. Votteri, H.P. Nii, J.S Aikins, L.M. Fagan, and E.A. Feigenbaum, 1978, "A physiological rule-based system for interpreting pulmonary function test results", in the *Proceedings of Computers in Critical Care and Pulmonary Medicine*, 375-379.

Kuo, B. and B. Konsynski, 1984, "Dialogue Management in DSS", *Management Information Systems*, University of Arizona, Tucson.

Langen, P.A., 1987, "Applications of artificial intelligence to simulation", in *Simulation Series, "Simulation and AI"*, Vol 18, No.3, 49-57, Luker and Birtwistle (editors).

Laski, J., 1965, "On time structure in (Monte Carlo) simulations", in *O.R. Quarterly*, Vol. 16, No.3, 329-339.

Laurie, P., 1985, "The Intelligent Database", in *Systems International*, August, pp 62-63.

Law, A.M. and J.S. Carson, 1979 , "A Sequential Procedure for Determining the Length of Steady State Simulation", *Operations Research*, Vol 27, No. 6, 1011-1025.

Law, A.M. and W.D. Kelton, 1982, "Confidence Intervals for Steady State Simulations, II: A Survey of Sequential Procedures, *Management Science, Vol.28*.

Lee, J.K. and E.G. Hurst Jr., 1983, "Solving semi-structured problems and the design of decision systems: Post model analysis", in a paper prepared for *Joint National Meeting of ORSA/TIMS*, University of Philadelphia, Orlando

Lehmann, A., 1987, "Expert systems for interactive simulation of computer system dynamics", in *Simulation Series*, Vol 18, No.3, 21-26, Luker and Birtwistle (editors).

Lehmann, A., B. Knodler, E Kwee and H Szczerbicka, 1986, "Dialog-oriented and knowledge-based modelling in a typical PC environment", in *Simulation Series, "Intelligent Simulation Environments"*, Vol 17, No.1, 133-138, Luker and Adelsberger (editors).

Lemmons, P., 1983, "Japan and the fifth generation", in *BYTE Publications Inc.*, November, pp 394-395.

Lenat, D.B., 1982a, "The nature of heuristics", in *Artificial Intelligence* Vol. 19, pp 189-249.

Lenat, D.B., 1982b, "EURISKO: a program that learns new heuristics and domain concepts", in *Artificial Intelligence*, No. 21

Lenat, D., R. Davis, J. Doyle, M. Genesereth, I. Goldstein and H. Schrobe, 1983, "Reasoning about reasoning", in *Building Expert Systems*, Hayes, Roth et al, Addison Wesley.

Leonard, T., 1979, "Why do we need significance levels?", from the *Mathematics Research Center Report*, University of Wisconsin.

Lindgren, B.W., 1960, *Statistical Theory* Third edition, Colliers MacMillan, International editions, New York.

Luker, P.A., 1986a, "Preface to the Proceedings of the Conference on Intelligent Simulation Environments", in *Simulation Series, "Intelligent Simulation Environments"*, Vol 17, No.1, p. viii, Luker and Adelsberger (editors).

Luker, P.A., 1986b, "Putting Expertise into Modeller", in *AI applied to Simulation. Simulation Series*, Vol 18, No.1, 103-105, Kerckhoffs, Vansteenkiste, Zeigler, (Editors). The Society for Computer Simulation, San Diego, Calif.

Luker, P.A., 1987, "Preface to the proceedings of the conference on simulation and AI", in *Simulation Series, "Simulation and AI"*, Vol. 18, No.3, p vii, Luker and Birtwistle (editors).

Luker, P.A. and H.H. Adelsberger (editors), 1986, the Proceedings of the Conference on Intelligent Simulation Environments, in *Simulation Series, "Intelligent Simulation Environments"*, Vol. 17, No. 1.

Luker, P.A. and G. Birtwistle (editors), 1987, the Proceedings of the Conference on AI and Simulation, in *Simulation Series, "Simulation and AI"*, Vol. 18, No. 3.

Maisel, H. and G. Gnugnoli, 1972, *Simulation of discrete stochastic systems*, Science Research Associates, Inc., U.S.A.

Mamrak, S.A. and P.D.Amer, 1980, "Estimating confidence intervals for simulations of computer systems", in *Simulation*, Vol. 35, No. 6, 199-205.

March, J.C. and H.A. Simon, 1958, *Organisations*, John Wiley & Sons, Inc., New York.

Marson, S., 1985, "Industrial Experimental Design", from an *MSc Simulation Assessment*, University of Warwick, Coventry.

Martin, G.R., 1984, "The overselling of expert systems", in *Datamation*, November, pp 76-80.

Martins, G.R., 1983, "Better Simulation Models for Decision Support", *The Rand Corporation*, Santa Monica, California.

Mathewson, S.C., 1974, "Simulation Program Generators", in *Simulation*, Vol. 23, No. 6, 181-189.

Mathewson, S.C., 1985, "Simulation Program Generators : Code and Animation on a PC", *Journal of the Operational Research Society*, Vol. 36, No. 7, 583-589.

Michaelsen, R. and D. Michie, 1983, "Expert systems in business", *Datamation*, November, pp 240-246.

Michie, D., 1980, "Expert Systems", in *The Computer Journal*, Vol. 23, No. 4, 369-376.

Middleton, S. and R. Zanconato, 1986, "BLOBS: An object-oriented language for simulation and reasoning", *Simulation Series, "AI applied to Simulation"*, Vol. 18, No.1, 130-135, Kerckhoffs, Vansteenkiste, Zeigler (editors).

Mintzberg, H., 1973, *The nature of managerial work*, pp 28-50, Harper Row, NY.

Mize and Cox, 1968, *Essentials of Simulation*, Prentice Hall.

Monro, D.M., 1982, *Fortran 77*, Edward Arnold, London.

Mood, A.M., F.A Graybill and D.C Boes, 1974, *Introduction to the theory of Statistics*, Third edition, International Student Edition, McGraw Hill.

Moreira de Silva, C.A.R., 1982, The development of a decision support system generator via action research, *PhD Thesis*, S.I.B.S., Warwick University, England.

Moreira de Silva, C. and J.M. Bastos, 1986, "The use of decision mechanisms in visual simulation for flexible manufacturing systems modelling", in *Simulation Series, "AI applied to Simulation"*, Vol. 18, No.1, 165-170, Kerckhoffs, Vansteenkiste, Zeigler (editors).

Moser, J.G., 1986, "Integration of artificial intelligence and simulation in a comprehensive decision support system", in *Simulation*, Vol. 47, No. 6, 223-229.

Muezelfeldt, R., A. Bundy, M. Uschold and D. Roberston, 1986, "ECO- an intelligent front end for ecological modelling", in *Simulation Series, "AI applied to Simulation"*, Vol. 18, No.1, 67-70, Kerckhoffs, Vansteenkiste, Zeigler (editors).

McCalla, G.I., L. Reid, P.F. Schneider, 1982, "Plan creation, plan execution and knowledge acquisition in a dynamic microworld", in *International Journal Man-Machine Studies*, Vol. 16, pp 89-112.

McCandless, T.P., 1986, "PDP mechanisms for intelligent display control", in *Simulation Series, "Intelligent Simulation Environments"*, Vol. 17, No.1, 87-91, Luker and Adelsberger (editors).

McDermott, D. and J. Doyle, 1980, "Non-Monotonic Logic 1", in *Artificial Intelligence* Vol. 13, pp 41-48.

McLaren, B., P.M. Neuss and O. de Groote, 1988, "The integrated modelling package (IMP): An object-oriented module for manufacturing simulation", in the *Proceedings of the European Simulation Multiconference, "Simulation Environments and Symbol and Number Processing on Multi and Array Processors"*, 231-236, Huntsinger, Karplus, Kerckhoffs and Vansteenkiste (editors), Simulation Council Inc.

McVicar, K. and H.R. Smith, 1988, "Knowledge-based simulation with frameworks", in the *Proceedings of the SCS Multiconference on Artificial Intelligence and Simulation : The Diversity of Applications*, (T. Henson, Ed.), 72-77, The Society for Computer Simulation International, San Diego, Calif.

Nance, R.E., 1984, "Model development revisited", from the proceedings of the *Winter Simulation Conference*, pp 75-80, Dallas.

Naylor, C., 1984, "How to build an inferencing engine", in *Expert Systems*, by R Forsyth; Chapman & Hall, London.

Neumann, G., 1986, "A prolog Tutorial", in *Simulation Series*, Vol. 17, No.1, 163-164, Luker and Adelsberger (editors).

Newkirk, R.T., 1988, "Simulation and statistics using personal portable microcomputers", in *Modelling and Simulation on Microcomputers*, pp 76-79, R.G. Lavery (editor).

Newkirk, R.T. and R.L. Walker, 1986, "System detected modelling tensions: System vs user intelligence", in *Simulation Series*, Vol. 17, No.1, 101-106, Luker and Adelsberger (editors).

Newman, R.A., 1982, "Statistics", in *Simulation Series, Computer Modelling and Simulation Principles of Good Practice*, Vol. 10, No. 2, 120-128, by J. McLeod.

Nielsen, N.R., 1987, "The impact of using AI techniques in a control system simulator", in *Simulation Series*, Vol. 18, No.3, 72-77, Luker and Birtwistle (editors).

Nozari, A., 1985, "Experimental designs in computer simulation", from the proceedings of the *Winter Simulation Conference*, pp 189, Gantz, Blais and Solomon (editors).

O'Keefe, R.M., 1984, "Developing simulation models : an interpreter for VIS", *PhD Thesis*, Southampton University.

O'Keefe, R.M., 1985, "Expert systems and operational research - mutual benefits", in *Journal of the Operational Research Society*, Vol. 36, No.2, 125-129.

O'Keefe, R.M., 1986a, "Simulation and expert Systems - A taxonomy and some examples", *Simulation*, Vol 46, No. 1, 10-16.

O'Keefe, R.M., 1986b, "Advisory Systems in Simulation", in *AI applied to Simulation. Simulation Series*, Vol 18, No.1, 73-78, Kerckhoffs, Vansteenkiste, Zeigler, Editors. The Society for Computer Simulation, San Diego, Calif.

O'Keefe, R.M., 1986c, "The three phase approach: a comment on 'strategy related characteristics of discrete event languages and models'", in *Simulation*, Vol. 47, No.5, 208-209.

O'Keefe, R.M. and R. Davies, 1983, *A microcomputer system for simulation modelling*, Paper presented at the Sixth European Congress on O.R..

O'Keefe, R.M. and J.W. Roach, 1987, "Artificial intelligence approaches to simulation", in *Journal of Operational Research Society*, Vol. 38, No.8, 713-722.

Oren, T.I., 1977, "Software for simulation of combined continuous and discrete systems: a state-of-the-art review", in *Simulation*, Vol. 28, No. 2, 33-45.

Oren, T.I., 1986a, "Knowledge bases for an advanced simulation environment", *Simulation Series, "Intelligent Simulation Environments"*, Vol. 17, No.1, 16-22, Luker and Adelsberger (editors).

Oren, T.I., 1986b, "Artificial intelligence and simulation", in *Simulation Series, "AI applied to Simulation"*, Vol. 18, No.1, 3-8, Kerckhoffs, Vansteenkiste, Zeigler (editors).

Oren, T.I., 1987a, "Artificial intelligence in modelling and simulation : Directions to explore", in *Simulation*, Vol. 48, No. 4, 131-134.

Oren, T.I., 1987b, "Quality assurance paradigms for artificial intelligence in modelling and simulation", in *Simulation*, Vol. 48, No.4, 149-151.

Oren T.I. and B.P. Zeigler, 1979, "Concepts for advanced simulation methodologies", in *Simulation*, Vol. 32, No. 3, 69-82.

Oren, T.I. and B.P. Zeigler, 1987, "Artificial intelligence in modelling and simulation: Directions to explore", in *Simulation*, Vol. 48, No. 4, 131-134.

Osman, I., 1986, "The brains behind Artificial Intelligence", in *the Sunday Times magazine*, July 20, pp 34-37.

Ostler, N. 1985, "Japan : a difference of emphasis", in *Systems International*, April, pp 69-70.

Paul, R.J. and S.T. Chew, 1987, "Simulation modelling using an interactive simulation program generator", in *Journal of Operational Research Society*, Vol. 38, No. 8, 735-752.

Paul, R.J. and G.J. Doukidis, 1986, "Further developments in the use of artificial intelligence techniques which formulate simulation problems", in *Journal of Operational Research Society*, Vol. 37, No. 8, 787-810.

Palme, J., 1977, "Moving pictures show simulation to user", in *Simulation*, Vol. 29, No. 6, 204-209.

Pave, A. and F. Rechenman, 1986, "Computer-aided modelling in biology on Artificial Intelligence approach", in *Simulation Series, "AI applied to Simulation"*, Vol. 18, No.1, 52-66, Kerckhoffs, Vansteenkiste, Zeigler (editors).

Peck, S.N., 1985, "Intermediate generality simulation software for production", in *Journal of Operational Research Society*, Vol. 36, No.7, 591-595.

Pegden, C.D., 1984, *Introduction to SIMAN with Version 2.0 Enhancements*, Systems Modelling Corporation, State College, Pennsylvania.

Pegden, L.A., T.I. Miles and G.A. Diaz, 1985, "Graphical interpretation of output illustrated by a siman manufacturing system simulation", from the proceedings of the *Winter Simulation Conference*, pp 244-251, Gantz, Blais, Soloman (editors).

Perez, J.C. and R Castanet, 1986, "Using cellular automata in graph theory modelling: A high performance solution to the HAMILTON problem", in *Simulation Series, "Intelligent Simulation Environments"*, Vol. 17, No.1, 9-15, Luker and Adelsberger (editors).

Phelps, R.I., 1986, "Artificial Intelligence - An overview of similarities with O.R.", in *Journal of Operational Research Society*, Vol. 37, No.1, 13-20.

Pidd, M., 1984, *Computer Simulation in Management Science*, Wiley, Great Britain.

Politakis, P. and S.M. Weiss, 1984, "Using empirical analysis to refine expert system knowledge bases", *Artificial Intelligence* Vol. 22, pp 23-48.

Poole, T.G., and Szymankiewicz, 1977, *Using Simulation to Solve Problems*, McGraw-Hill, London.

Pope, S.T., 1985, "Unix and A.I.", in *Systems International*, April, pp 64-66.

Pritsker, A.A.B., 1984, *Introduction to Simulation and SLAM II*, Systems Publishing Corporation, Indiana.

Quinlan, J.R., 1983, "Inferno : a cautious approach to uncertain inference", in *The Computer Journal*, Vol. 26, No.3, 255-269.

Rada, R., 1984, "Automating knowledge acquisition", in *Expert Systems*, by Richard Forsyth; Chapman & Hall, London.

Radiya, A. and R.G. Sargent, 1987, "Logic programming and discrete event simulation", in *Simulation Series, "Simulation and AI"*, Vol. 18, No.3, 64-71, Luker and Birtwistle (editors).

Radzikowski, P., 1983, "Perspectives on the business decision support expert systems", *TIMS/ORSA meeting presentation* School of Business, Seton Hall, University South Orange.

Rajagopalan, R., 1986, "Qualitative modelling and simulation", in *Simulation Series, "AI applied to Simulation"*, Vol. 18, No.1, 9-26, Kerckhoffs, Vansteenkiste, Zeigler (editors).

Ranyard, J.C., and R.A. Blewitt, 1983, *O.R. and personal computing*, Paper presented at the Sixth European Congress on O.R..

Rao, M.J. and R.D. Sargent, 1988, "An experimental advisory system for operational validity ", in the *Proceedings of the SCS Multiconference on Artificial Intelligence and Simulation : The Diversity of Applications*, (T. Henson, Ed.),245-250, The Society for Computer Simulation International, San Diego, Calif.

Reddy, R., 1987, "Epistemology of knowledge based simulation", in *Simulation*, Vol. 48, No.4, 162-166.

Reddy, Y.V., M.S. Fox and N.Husain, 1985, "Automating the analysis of simulations in KBS", in *A.I., Graphics and Simulation*, pp 34-40, G. Birtwistle (editor).

Rich, E., 1983, *Artificial Intelligence*, McGraw Hill.

Rietz, F., 1983, "Cheaper, better and quicker - warehouse design using visual interactive simulation", *ISTEL Ltd.*.

Robertson P., 1986, "A rule based expert simulation environment", in *Simulation Series, "Intelligent Simulation Environments"*, Vol. 17, No.1, 9-15, Luker and Adelsberger (editors).

Rockart, J.F. and A.D. Gescenzi, 1983, *"A process for the rapid development of systems in support of managerial decision-making"*, M.I.T.

Roussel, P., 1975 *Prolog: Manuel de Reference et d'Utilisation*, Groupe d'Intelligence Artificielle, U.E.R de Luminy, Universite d'Aix-Marseilles II.

Rozenblit, J.W. and B.P. Zeigler, 1985, "Concepts for Knowledge-Based System Design Environments", in *Proceedings of the 1985 Winter Simulation Conference*, 223-231, I.E.E.E.

Rubens, G.T., 1979, A study of the use of V.I.S. for decision making in a complex production system, *MSc Thesis*, S.I.B.S., Warwick University.

Ruiz-Mier, S., J. Talavage, D. Ben-Arieh, 1985, "Towards a Knowledge-Based Network Simulation Environment" in the *Proceedings of the 1985 Winter Simulation Conference*, 232-235, I.E.E.E.

Sargent, M. and R.L. Shoemaker, 1984, *The IBM PC From the Inside Out*, Addison Wesley.

Schriber, T., 1974, *Simulation using GPSS*, John Wiley.

Schruben, L., 1982, "Detecting Initialisation Bias in Simulation Output", *Operations Research*, Vol 30, No. 3, 569-590.

Schruben, L. and D. Goldsman, 1982, "Using process central limit theorems in analyzing simulation output", in the *Proceedings of the 1982 Winter Simulation Conference*, pp 599-602, Highland, Chao and Madrigal (editors), I.E.E.E.

Schruben, L., H. Singh and L. Tierney, 1983, "Optimal tests for initialisation bias in simulation output", in *Operations Research*, Vol. 31, No. 6, 1167-1178.

Schwarz, G., 1978, "Estimating the dimension of a model", in *Annals of Statistics*, Vol. 6, 461-464.

Secker, R.J.R., 1977, "That V.I.S. offers a viable technique for examining production planning and scheduling problems, *MSc Thesis*, S.I.B.S., Warwick University, England.

Secker, R.J.R., 1979, "Visual Interactive Simulation Using a Mini-Computer", in *Journal of Operational Research Society*, Vol. 30, 379-381.

Seila, A.F., 1984, "Multivariate output analysis in simulation: the state of the art", from the proceedings of the *Winter Simulation Conference*, pp 283, S. Sheppard, U. Pooch, D.

Pegden (editors).

Sena, J.A. and L.M. Smith, 1987, "An expert system for the controller", in *Simulation Series, "Simulation and AI"*, Vol. 18, No.3, 27-32, Luker and Birtwistle (editors).

Senge, P.M., 1977, "Statistical estimation of feedback models", in *Simulation*, Vol. 28, No.6, 177-184.

Shannon, R.E. 1984, "Artificial Intelligence and Simulation", keynote address in the proceedings of the *Winter Simulation Conference*, pp 3-9, S.Sheppard, U. Pooch, D. Pegden (editors).

Shannon, R.E., 1986, "Intelligent Simulation Environments", in *Intelligent Simulation Environments. Simulation Series*, Vol 17, No.1, 150-156, Luker and Adelsberger, Editors. The Society for Computer Simulation, San Diego, Calif.

Shannon, R.E.; R.Mayer ; H. H. Adelsberger, 1985, "Expert Systems and Simulation", *Simulation*, Vol 44, No.6, 275-284.

Shaw M.L.G., 1987, "Knowledge support systems", in *Simulation Series, "Simulation and AI"*, Vol. 18, No.3, 58-63, Luker and Birtwistle (editors).

Shaw, M.L.G. and B.R. Gaines, 1986, "A framework for knowledge-based systems unifying expert systems and simulation", in *Simulation Series, "Intelligent Simulation Environments"*, Vol. 17, No.1, 38-43, Luker and Adelsberger (editors).

Shortliffe, E.H., 1976, *Computer based medical consultations : MYCIN*, Elsevier Publishing Co.

Shortliffe, E.H., 1987, "Computer programs to support clinical decision making" in *JAMA*, Vol. 258, No. 1, 61-67.

Simon, H.A., 1960, *The New Science of Management Decision*, Harper and Row.

Smith, S.F., 1984, "Adaptive Learning Systems", in *Expert Systems*, by Richard Forsyth; Chapman & Hall, London.

Smith, V.L., 1986, "Visual Interactive Modelling - Letters and Viewpoints", *Journal of the Operational Research Society*, Vol 37, 1017-1021.

Soetarman, B. and A.H. Bond, 1988, "Multiple abstraction in knowledge-based simulation", in the *Proceedings of the SCS Multiconference on Artificial Intelligence and Simulation : The Diversity of Applications*, (T. Henson, Ed.), 61-66, The Society for Computer Simulation International, San Diego, Calif.

Solomon, S.L., 1982, "Simstat-A simple statistical package to support simulation", from the proceedings of the *Winter Simulation Conference*, pp 307-311, Highland, Chao and Madrigal (editors).

Solomon, S.L., 1983, *Simulation of Waiting-Line Systems*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Soroka, B.T., 1979, *"What can't robot languages do? "*, University of Southern California.

Spinelli de Carvallo, R. and J.G. Crookes, 1976, "Cellular Simulation", in *O.R. Quarterly*, Vol. 27, No.1, 31-40.

Sprague, R. and E. Carlson, 1982, *Building effective decision support system*, Prentice-Hall Inc., Englewood Cliffs, NJ.

Sprent, P., 1977 *Statistics in Action* Penguin Books, England.

Stallman, R.M. and G.J. Sussman, 1977, "Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis", in *Artificial Intelligence*, No. 11, 85-114.

Standridge, C.R., J.R. Hoffman, D.K. LaVal, 1985, "Presenting simulation results with TESS graphics", from the proceedings of the *Winter Simulation Conference*, pp 237-243, Gantz, Blais and Soloman (editors).

Stefik, M., J. Aikins, R. Balzer, J. Benoit, L. Birnbawn, F. Hayes-Roth and E. Sacerdoti, 1982, "The organisation of expert systems : a tutorial", *Artificial Intelligence* Vol. 18, pp 135-173.

Stephenson, J., 1982, "The role of simulation in education", in the *Proceedings of the 1982 Winter Simulation Conference*, pp 420-426

Stevens, A., 1984, "How shall we judge an expert system? ", in *Expert Systems*, by Richard Forsyth; Chapman & Hall, London.

Stevens, G., 1982, "O.R. worker, Information systems analysis and the challenge of the micro", in *Journal of Operational Research Society*, Vol. 33, 921-929.

Stevens, G., 1983, "User friendly computer systems? A critical examination of the concept", in *Behaviour and Information Technology*, Vol. 2 No. 1 3-16.

Stewart, S.D. and G. Watson, 1985, "Applications of artificial intelligence", in *Simulation*, 306-310.

Stoner, G., 1985, "Expert systems: Jargon or challenge", in *Accountancy*, February, pp 142-145.

Stonier, T., 1984, "The knowledge industry", *Expert Systems*, Chapman and Hall, London.

Swaan Arons, de H., E.P. Jansen and P.J.T. Lucas, 1986, "Building and constructing expert systems in simulation with DELFI-2", in *Simulation Series, "AI applied to Simulation"*, Vol. 18, No.1, 111-113, Kerckhoffs, Vansteenkiste, Zeigler (editors).

Tate, P., 1984, "Europe's expert experience", in *Datamation*, February, pp 198-199.

Taylor, R.P., 1985, "Simulation Study of an Automated Die Shop", *MSc Thesis*, University of Warwick, England.

Taylor, R.P., 1988, *WES - Technical Documentation*, S.I.B.S., University of Warwick, England.

Taylor, R.P. and R.D. Hurrion, 1988, "An expert advisor for simulation experimental design and analysis", in the *Proceedings of the SCS Multiconference on Artificial Intelligence and Simulation : The Diversity of Applications*, (T. Henson, Ed.), 238-244, The Society for Computer Simulation International, San Diego, Calif.

Thorngate, W., 1980, "Efficient decision heuristics", in *Behavioral Science*, Vol. 25, pp 219-225.

Tocher, K.D., 1963, *The Art of Simulation*, English Universities Press, London.

Tocher, K.D., 1965, "Review of Simulation Languages", in *O.R. Quarterly*, Vol. 16, No.2, 189-217.

Treleaven, P. and I. Lima, 1982, "Japan's Fifth Generation Computer Systems", in *Computer*, August, pp 79-88.

Turnquist, M.A. and J.M. Sussman, 1977, "Toward guidelines for designing experiments in queuing simulation", in *Simulation*, Vol. 28, No.5, 137-144.

Tversky, A. and D. Kahneman, 1974, "Judgement under uncertainty: Heuristics and biases", in *Science*, No. 185, 1124-1131.

Unger, B., A. Dewaer, J. Cleary and G Birtwistle, 1986a, "A distributed software prototyping and simulation environment: JADE", in *Simulation Series, "Intelligent Simulation Environments"*, Vol. 17, No.1, 63-71, Luker and Adelsberger (editors).

Unger, B., A. Dewaer, J Cleary and G. Birtwistle, 1986b, "The Jade approach to distributed software development", in *Simulation Series, "AI applied to Simulation"*, Vol. 18, No.1, 178-188, Kerckhoffs, Vansteenkiste, Zeigler (editors).

Vaucher, J.G., 1985, "Views of modelling: Comparing the simulation and A.I. approaches", in *A.I., Graphics and Simulation*, pp 3-7, Birtwistle (editor).

Vaucher, J.G. and G. Lapalme, 1987, "Process oriented simulation in Prolog", in *Simulation Series, "Simulation and AI"*, Vol. 18, No.3, 41-46, Luker and Birtwistle (editors).

Vazsonyian, A., 1980, "Predicate calculus and database management systems", *Interfaces* Vol. 10, No. 1, 64-71.

Verity, J.W., 1984, "A.I. tools arrive in force", in *Datamation*, November, pp 44-53.

Vinod, B., 1986, "Exponential queues with server vacations", *Journal of Operational Research Society*, Vol 37, No.10, 1007-1014.

Wahl, D., 1986, "An application of declarative modelling to aircraft fault isolation and diagnosis", in *Simulation Series, "Intelligent Simulation Environments"*, Vol. 17, No.1, 25-28, Luker and Adelsberger (editors).

Wales, F.J. and P.A. Luker, 1986, "An environment for discrete event simulation", in *Simulation Series, "Intelligent Simulation Environments"*, Vol. 17, No.1, 58-62, Luker and Adelsberger (editors).

Walker, T.C., R.K. Miller, 1986, *Expert Systems 1986: An Assessment of Technology Applications*, SEAI Technical Publications.

Walsham, G., 1980, "A computer simulation model applied to design problems of a petrochemical complex", in *Successful O.R.*, by R. Colcutt, The O.R. Society.

West, D., G. Lemow and B. Unger, 1987, "Optimising time warp using the semantics of abstract data types", in *Simulation Series, "Simulation and AI"*, Vol. 18, No.3, 3-8, Luker and Birtwistle (editors).

Wilkins, D., 1982, "Using knowledge to control tree searching", in *Artificial Intelligence*, Vol. 18, pp 45-51.

Wilson, J.R., 1982, "Variance reduction techniques", in the *Proceedings of the 1982 Winter Simulation Conference*, pp 605-612, Highland, Chao and Madrigal (editors), I.E.E.E.

Winston, P.H., 1977, *Artificial Intelligence*, Addison Wesley.

Winston, P.H., 1982, "Learning New Principles from Precedents and Exercises", in *Artificial Intelligence*, Vol. 19, pp 321-350.

Withers, S.J., 1981, Towards the on-line development of visual interactive simulation models, *PhD Thesis*, S.I.B.S., Warwick University.

Witte, T. and R. Grzybowski, 1988, "Object-Based Simulation: Foundations and Implementation in Modula 2", in the *Proceedings of the European Simulation Multiconference, "Simulation Environments and Symbol and Number Processing on Multi and Array Processors"*, 193-198, Huntsinger, Karplus, Kerckhoffs and Vansteenkiste (editors), Simulation Council Inc.

Xining L. and B. Unger, 1986, "Predicting X-Tree network performance using the Jade environment", in *Simulation Series, "Intelligent Simulation Environments"*, Vol. 17, No.1, 75-79, Luker and Adelsberger (editors).

Xining, L. and B. Unger, 1987, "Languages for distributed simulation", in *Simulation Series, "Simulation and AI"*, Vol. 18, No.3, 35-40, Luker and Birtwistle (editors).

Xiong, G., and A Song, 1986, "An expert system for dynamic system simulation", in *Simulation Series, "AI applied to Simulation"*, Vol. 18, No.1, 106-110, Kerckhoffs, Vansteenkiste, Zeigler (editors).

Yazdani, M., 1984, "Knowledge engineering in PROLOG", in *Expert Systems*, by Richard Forsyth; Chapman & Hall, London.

Young, R.M., 1979, "Production systems for modelling human cognition", in *Expert systems in the micro-electronic age*, Michie (editor).

Zalevsky, P.A., 1988, "Knowledge-based simulation of manufacturing facilities", in the *Proceedings of the SCS Multiconference on Artificial Intelligence and Simulation : The Diversity of Applications*, (T. Henson, Ed.), 67-71, The Society for Computer Simulation International, San Diego, Calif.

Zanakis, S.H. and J.R. Evans, 1981, "Heuristics- 'Optimisation' : why, when and how to use it", in *Interfaces*, Vol. 11, No.5, 84-89.

Zeigler, B.P., 1976, *Theory of Modelling and Simulation*, John Wiley & Sons.

Zeigler, B.P. and L. de Waal, 1986, "Towards a knowledge-based implementation of multifaceted modelling methodology", in *Simulation Series, "AI applied to Simulation"*, Vol. 18, No.1, 42-51, Kerckhoffs, Vansteenkiste, Zeigler (editors).

XXX, 1984, "*Savoir, Technical Description*", ISI Ltd.,

XXX, 1984, "Counsellor: a SAVOIR case history", Press release from *ISI Ltd.*, September 19.

XXX, 1984, "Expert Systems. What they are and why they are needed", Press release from *ISI Ltd.*, September 19.

XXX, 1984, "Savoir, the powerful flexible expert system for general application", press release from *ISI Ltd.*, September 19.

XXX, 1985, "*Expert Case. A technical overview*", ESI

XXX, 1985, "The ESPRIT programme 1985 fact sheet", in *Alvey News*, February, I.E.E.

Appendix 1

Programming in Prolog

Introduction

This appendix gives a brief outline of the concepts behind programming in Prolog. This outline is drawn from Adelsberger (1984) and Flitman (1986). The standard Prolog as described by Clocksin and Mellish (1981) is used.

Computer programming in Prolog consists of:

- declaring facts about objects and their relationships;
- defining rules about objects and their relationships;
- asking questions about objects and their relationships.

Facts

For example, to say that "Mozart composed Don Giovanni" states that a relationship links two objects. This could be written in Prolog in standard form:

```
composed(mozart, don_giovanni).
```

The name of the relationship is given first, and the objects are separated by commas and are enclosed in parentheses.

A collection of facts (and later rules) is called a database. For example,

```
composed(mozart, don_giovanni).
composed(verdi, rigoletto).
composed(verdi, macbeth).
composed(rossini, guglielmo_tell).
```

Questions and Variables

It is possible to ask questions in Prolog. Two different types of questions can be asked:

is-question (eg "did Mozart compose Don-Giovanni?"), and

which-questions (eg "who composed Don-Giovanni?")

For "is-questions", the answer is 'yes' or 'no'. In the above question the answer would be 'yes'. For "which-questions", one has to specify one or more variables. 'X' is the variable in the above example, and the result would be:

X = mozart

If there are more solutions to a question as in "which operas where composed by Verdi?"

?-composed(verdi,X).

all solutions are listed:

X = rigoletto X = macbeth

When Prolog is asked a question containing a variable, it searches through all its facts to find an object that the variable could stand for.

Syntax

Prolog programs consist of terms. A term is a constant, a variable or a compound term (structure). Constants are numbers or atoms. Names of atoms begin with a lower case letter. Variables are always capitalised. A structure is written by specifying its functor ('composed' in the above example), followed by its components (also called arguments) enclosed in parentheses, separated by commas. Lists are a special form of compound term.

Conjunctions

Given the following database:

likes(mary,food).

```
likes(mary,wine).

likes(john,wine).

likes(john,mary).
```

One could ask in Prolog :"Is there anything that John and Mary both like?" in the following form:

```
?-likes(mary,X),likes(john,X).
```

The comma is pronounced 'and', and expresses the fact that one is interested in the conjunction of these two goals.

Rules

A rule is a general statement about objects and their relationships. Rules are used to say that a fact depends on a group of other facts. For example, to say that a person is someone's sister, one would say:

X is female and

X and Y have the same parents'

In Prolog syntax, one would write :

```
sister_of(X,Y) :-

    female(X),

    parents(X,Z1,Z2),

    parents(Y,Z1,Z2).
```

The symbol ':-' is pronounced 'if'.

Lists

A list is an ordered sequence of elements that can have any length. Lists are written in Prolog using square brackets. Elements are separated by commas as in:

```
languages([gpss, simscript, simula, slam]).
```

Some other lists are:

```
[]

[[the,[boy]],[kicked],[the,[ball]]]
```

The first list is the empty list. The second list represents the grammatical structure of a simple sentence. A vertical bar is used to split a list into its head and tail.

```
?-language([X|Y]).

    X = gpss

    Y = [simscript, simula, slam]
```

Recursion

Recursion is a powerful technique used to express complex algorithms and structure in an easy way. In many cases algorithms can be expressed in two different forms. The first uses recursion. The second uses loops. A simple example is the computation of a factorial function. In Prolog, recursion is the normal and natural way.

The membership test for an element of a list is a simple demonstration of recursion in Prolog:

```
member(X,[X|Y]).

member(X,[H,Y]) :- member(X,Y).
```

This can be read as:

The element given as the first argument is a member of the list given as the second argument, if the list starts with the element (the fact in the first line) *or if the element is a member of the tail* (the rule in the second line).

Possible questions are:

?-member(d,[a,b,c,d]).

yes

?-member(e,[a,b,c,d]).

no

It is possible to get all members of a list by asking: ?-member(X,[a,b,c,d]).

X = a

X = b

X = c

X = d

Appendix 2

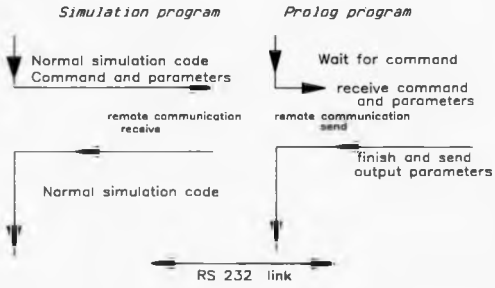
Fortran-Prolog Interprocessor RS232 Serial Link

This appendix provides an overview of the link operations as developed by Dr Andrew Flitman. These notes are based on Flitman's working notes (1984-1986) and the author's own transcripts that were written during the conversion of Flitman's working link to the controller application. The main impact of the author's work is found in appendix 3.a

Communications via Asynchronous Adaptor

The main reference is from Sargent & Schoemaker, (chapter 10, pages 355 to 396, 1984). The link via a RS232 data link is achieved using a modified version of the standard "S/"Q protocol (a.k.a. DC1/DC3 or XON/XOFF) for handshaking.

An example:



The Prolog waits to receive a command from the simulation. The command consists of a predicate (name) followed by input parameters. This command is then executed by Prolog, while the Fortran simulation awaits its completion.

During "command" execution, various forms of interactions between Fortran and Prolog may take place..

When all interactions for the "command" are complete, the Prolog will inform the Fortran.

How does it work ?

The Asynchronous Adaptor at both computers need to be set at the same communications parameters:

9600 baud (960 characters/second).

1 stop bit

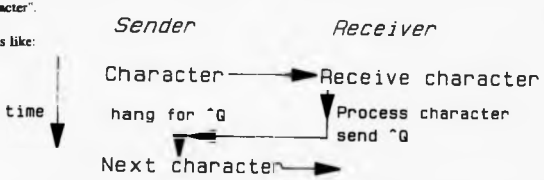
no parity

8 bit words

When characters are being sent at this speed, there is a danger that the receiver is not processing them fast enough : in which case some characters may be lost. To overcome this, a simple version of the ^Q/^S protocol is used.

The idea is as follows. When the sender passes a character to the receiver, no more characters are passed across until the receiver echoes back a ^Q indicating that it is "ready to receive the next character".

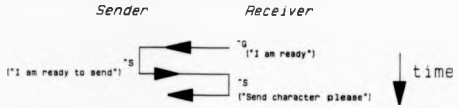
This looks like:



Note that in a normal ^Q/^S protocol, the receiver sends a ^S to stop the sender. Here ^S is implicit after each character is sent: Flitman called this the "implicit ^S protocol"

The "implicit ^S protocol" is not the only form of handshaking used. It has also proved necessary to coordinate the two communicating programs so that they are ready to send/receive at

precisely the correct moment. Flitman called this the "coordinating protocol".



The Prolog Communications Protocol

Prolog programs operate on a variety of data structures. In order for a simulation Fortran program to interrogate a Prolog program, it is necessary for Fortran to send commands and parameters to the Prolog. The commands would be the name of a Prolog predicate. The parameters may be integers, atom names or lists, and must be sent in the correct order.

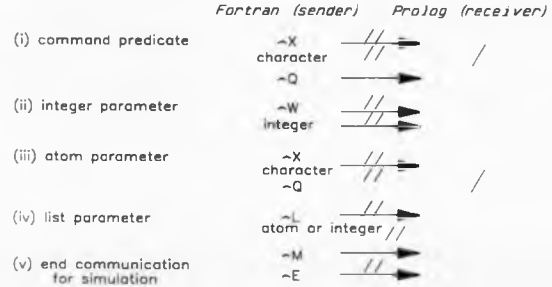
The Prolog has the ability to request the user of the Fortran for an input list, integers (<256) or atoms, as well as printing on the Fortran terminal any message (question, integer (<2356), list or atom).

Each of these communication options (including the ones starting/ending a predicate execution) have their own distinguishing control signals. Schematically, this looks like:

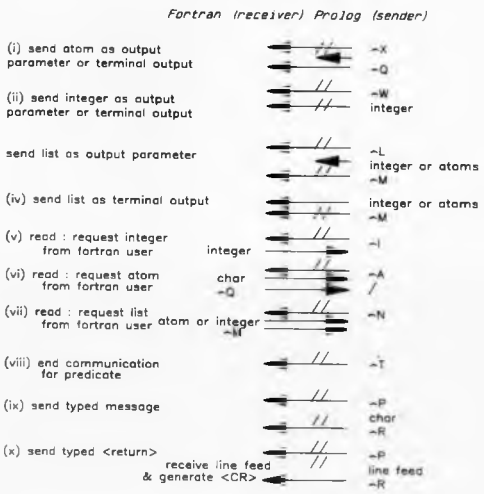
// : coordinating protocol

/ : implicit ^S protocol

A. Communications from Fortran to Prolog



B. Communications from Prolog to Fortran



Prolog Communications Predicates

To simplify the use of the link, some simple Prolog predicates have been written and stored.

Reserved words

command, comcall, reada, commsend, remote_list_request,
commend, commsg, outputr, readri, readrl,
writera, writeri, nlr, men, comlsend, comlsnd,
readi, remote_list, copmlrec, smth, retractb, comati, rev, append

Limitations

- No reals can be sent.

- Only integers less than 256 can be sent.

- "Command" predicate ("wait for a command and execute it) is not responsive to end of simulation.

- No uniform user interface. Because the remote read_list is responded to in Prolog and not MACRO86, the user must answer to a request for an atom or integer with a "<CR>". However for a list, he must reply with a ".<CR>" as in Prolog.

The Fortran-Prolog Interprocessor link

The Assembler

Communication is achieved via a common block in Fortran:

```
INTEGER*1 char(120)
common/ata/char
```

This can be also accessed by the assembler, via a public segment ata

```
ATA SEGMENT BYTE COMMON 'DATA'
CHAR DB 120 DUP(?)
ATA ENDS
```

All other data elements in the assembler are contained within another segment. These data definitions are mainly there to communicate with the Prolog expert at the other end of the link.

Assembler modified front end

The aim of the modified front end is to convert the calling of the assembler routines, in order to make it natural in Fortran.

Prolog's access to assembler code is achieved by having 51 bytes common to both sets of code. In these 51 bytes, provision is given for:

(i) an operation code number (OP) which dictates the section of assembler that PROLOG is requesting.

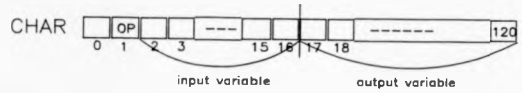
(ii) 8 input variables, which may be either atoms or integers. The Prolog link has been written so as to need only one such input variable.

(iii) 8 output variables, which may be either atoms or integers. The Prolog link has been written so as to need only one such output variable.

In order to make this link usable by Fortran, a front-end has been written to the assembler which allows the Fortran to specify operation number and input variables and a back end to receive output variables via the common data area ATA

ATA is full of CHAR

CHAR is a byte array (integer*1)



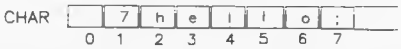
Input variable

(i) Integers (8bit)

These are required by operation codes 3 & 8. The front end checks the OP number (placed in CHAR(1)) against these values and if found then the integer is deemed to be the value in CHAR(2). All other elements of the CHAR array are zero. The front-end places the OP value and integer value into areas allocated for them by a Prolog Program. The link is then performed in the way normal in Prolog (after branching the code indicated by OP).

ii) Atoms

These are required by operation code number 7. If this OP code is found in CHAR(1), the front end deems the atom to be the characters in CHAR(2), CHAR(3), etc with the atom being delimited by a semi-colon. For example, the atom 'hello' is represented by



This atom is then processed by the front-end to have the same form as a prolog atom. The link is then performed in the normal way to Prolog (after branching to the code indicated by OP).

(iii) No input variable

The front-end here simply branches to the correct area of code, as indicated by OP (CHAR(1)).

Assembler modified back end

This modifies the output variables to a Prolog routine to make them usable by Fortran. Output integers are placed in CHAR(17) and output atoms are placed in CHAT(17),CHAR(18)... terminating with a semi-colon. As in the Prolog end of the link, certain other forms of output are possible to indicate:

(i) request for user to type a list,

(ii) end of list.

For these, the output integer is set to

(i) 50 (at a point where normally a zero is expected,

(ii) -1

These values act as flags which are tested by the calling of independent Fortran routines.

Fortran Routines

Problem Independent

These fall into two categories:

(i) those that have their equivalent in Prolog's link;

(ii) those designed to make the link easier to use (eg data structures, etc).

(i) Those that have a Prolog equivalent

Prolog	Fortran
writera(A)	wrra
writeri(I)	wrri
reada(A)	rda
readi(I)	rdi
commend	cend
comsend(X)	csend
external_code_(16,C1,C7)	ends
read_remote_request	rerq
comlsnd(L)	lsnd(L)
rev(X,Y)	revl(L)
comlsend(L)	lsend
comlrec(L)	rdl

(ii) Those designed to make the link easier

These allow the easy manipulations of data structures. The sending of parameters, and the simplification of command sending.

a The simplification of command-sending

Because of the range of commands to be sent in any application is finite, the atoms for each command can be given an index number. To send a command, all one need do is supply the index to the subroutine CFILL and then call the send command subroutine CSEND.

CFILL is a problem dependent routine. CFILL is used as follows:

```
SUBROUTINE CFILL(IMESS)
      integer*2 IMESS
      call store(imess,1,'hello;')   1 is the index of hello
      call store(imess,2,'goodbye;')  2 is the index of goodbye
RETURN
END
```

To call the command "hello", the simulation need only incorporate:

```
CALL CFILL(1)
CALL CSEND
```

b. The simplification of list construction

Similarly, a problem dependent PFILL helps the construction of lists. For example, assume the following PFILL.

```
SUBROUTINE PFILL(IMESS,P)
      integer*2 IMESS,P
      call store(imess,1,'chelsea;')
      call store(imess,2,'will;')
      call store(imess,3,'win;')
RETURN
END
```

Then to send the Prolog expert the list [chelsea,1,2,win], the simulation must include the following code:

```
Plist(1) = -1
Plist(2) = 1
Plist(3) = 2
Plist(4) = -2
Plist(5) = -100
call PFILL(1,1)
call PFILL(3,2)
```

PFILL(m,n): fill array row n with atom index number m.

(iii) Sending parameters

Parameters can be sent before they are needed. A problem dependent subroutine needs to be written to indicate what the parameters are and to send them.

(a). Storing of parameters prior to being sent.

This version of the link allows the storage of

- 1 list

- 3 atoms

- 20 integers

The integers are stored in the integer*1 array intp(20). An atom is stored in one of the 3 integer*1 arrays: atm1(20), atm2(20), atm3(20). To store an atom in one of these, the simulation must

CALL FATM(N,ATOM) N = 1,2,3 indicating atm1, atm2, atm3

Atom is the holerith form of atom:

eg call FATM(1,'ricky;')

(b) Sending parameters

This is achieved via a problem dependent subroutine that checks the name of the command just sent and then sends off the respective parameters, having picked them up from storage. This is done in a subroutine RPRAMS.

(iv) Other user-called routines

CLRC empties the common ATA block

(v) Other routines

These are used internally by the user-called Fortran subroutines.

Appendix 3

Modifications to Source Code of Software Used

The work of this thesis borrowed from two sources of programs. The first is the MICROVISION visual simulation package. The second supports the RS232 serial link for Prolog/Fortran communications. The implementation of the intelligent controller required some modifications to source code. These changes are given in this appendix.

Appendix 3 - a : Modifications to the RS232 serial link

In order to use the link developed by Flitman to control large simulation models, it has been necessary to alter the storage allocation declaration in his assembler files. This appendix presents the changes that are necessary to Flitman's code in order to use the intelligent controller programs. The complete original code may be found in Flitman (1986).

File PRLNK2.ASM must be changed from :

```
        NAME    LNK
CGROUP GROUP   CODE
        PUBLIC  prlink
DGROUP GROUP   ATA,KINSKJ
        INCLUDE DAT.ASM
CODE    SEGMENT     BYTE    PUBLIC  'CODE'
        ASSUME CS:CGROUP,DS:DGROUP

init    macro
        mov  ax,0e3h            ;set up for 9600 baud,1 stop bit,no
        int  14h          ;parity, 8-bit words
        ...
```

to the new code:

```
        NAME    LNK
DGROUP  GROUP   @C,@D,KINSKI
        public prlink

@C      SEGMENT WORD    PUBLIC  'DATA'
PRLINK  DD      _PRLINK
@C      ENDS

@D      SEGMENT WORD    PUBLIC  'DATA'
PCHAR   DD      CHAR
@D      ENDS

        INCLUDE DAT.ASM

?PRLINK SEGMENT BYTE    PUBLIC  'CODE'
        ASSUME  CS:?PRLINK,DS:DGROUP

init    macro
        mov     ax,0e3h         ;set up for 9600 baud,1 stop bit,no
        int     14h             ;parity, 8-bit words
        ...
```

File DAT.ASM must be changed from:

```
ATA     SEGMENT BYTE    COMMON 'DATA'
CHAR    DB      120 DUP(?)
ATA     ENDS

KINSKI  SEGMENT BYTE    'DATA'
op      db      0
incnt   db      0
outcnt  db      0
        ...
KINSKI  ENDS
```

to the code:

```
ATA     SEGMENT WORD    COMMON 'EXTRA'
CHAR    DB      120 DUP(?)
ATA     ENDS

KINSKI  SEGMENT WORD    'DATA'
op      db      0
incnt   db      0
outcnt  db      0
        ...
KINSKI  ENDS
```

Appendix 3.b : Modifications to the MICROVISION package

In order to use the MICROVISION package with the CONTROLLER, it has been necessary to change some of the source code. The MICROVISION package is made up of a number of modules. Module SYS1.FOR was altered to permit:

- automatic execution of experiments without keyboard input,
- monitoring of queue utilisations.

In this appendix, the subroutines that require changes are listed.

```
      cr50
      SUBROUTINE ADVANC(IEVENT,ISIM,IENT)
            integer*4 it,isim,isimt,itime
            integer ifp(2),ipd
           INTEGER TIMSET
           INCLUDE 'VSIM'
*****            common/exst/ifp,ipd
*****       if ((iprol.ne.0).and.(imp.eq.0)) goto 997
     42    continue
     50    TIMSET=IARR(10)
           IARR(15)=0
           ITIME=iarr(1)
           K=ISIZE(TIMSET)
            if(k.le.1)call dsperr(50,50,index)
            if(k.le.1)goto 990
           IPRERF=TIMSET-11
           ITAILN=TIMSET-11
           DO 100 I=1,K
           IREF=IARR(IPRERF)
           IF(IREF.EQ.ITAILN) GO TO 990
           IENT=IARR(IREF-1)
                ip=iarr(ient)
           ISIMT=iarr(ip)
           IF(ISIMT.GE.ITIME) GO TO 200
              call dsperr(50,52,index
    100   IPRERF=IREF
           GO TO 990
```

```
  200  CALL INTERV
       IF(IARR(15).EQ.1) GO TO 50
       IF(IARR(12).EQ.1) GO TO 980
       IF(ISIMT.EQ.ITIME) GO TO 400
       ITIME=ITIME+1
       DO 300 it=ITIME,ISIMT,1
       IF(IARR(6).EQ.0) GO TO 316
            J=IARR(55)*10
             if(j.le.0)goto 316
       DO 25 II=1,J
        r1=9999.9*ii
   25  R1=SQRT(r1)
  316  CONTINUE
       CALL FORMTI(it)
       tarr(1)=it
       CALL INTERV
       IF(IARR(15).EQ.1) GO TO 50
       IF(IARR(12).EQ.1) GO TO 980
  300  CONTINUE
  400  ITEM=IREF-1
       CALL DELETE(ITEM,TIMSET)
       ISIM=ISIMT
       tarr(1)=ISIM
       IEVENT=IARR(IENT-1)
        if(ievent.eq.-3)ievent=999
       IF(IEVENT.GT.0) GO TO 990
       IF(IEVENT.EQ.-1) IARR(13)=1
       IF(IEVENT.EQ.-1) GO TO 50
       IF (IEVENT.LT.-2) GO TO 50
       IARR(6)=1
       IARR(13)=1
       CALL REFORM
       GO TO 50
  980  IEVENT=0
       ISIM=tarr(1)
       IENT=0
       IARR(12)=0
  990    if(iarr(20).eq.0)goto 995
         call ctrace(ievent,isim,ient)
  995  if(d(19).eq.0)goto 999
         call vdisps(timset)
         call blanks(timset,8)
         goto 999

****  997     call vclass(ifp,2,'$',19,76)
****          call vset(ipd,'$',19,1,1,1,0,23)
****          call vaddla(ifp(1),ipd)
****          if (iprol.eq.4) call schedl(999,ibatim,ihead(ipd))
****          if (iprol.eq.2) call schedl(999,igotim,ihead(ipd))
****          imp = 1
****          goto 42
       999    return
              end

       c
```

```
      C
      SUBROUTINE INTERV
            integer*4 it,itemp
            integer*2 itime
           INTEGER*2 IAR(8),ival,i
           INCLUDE 'VSIM'
      10   CONTINUE
           IF(IARR(13).EQ.1) GO TO 50
      C     ALL INTERA(IARR(13))
           IF (IARR(13).EQ.0) GO TO 990
      50   IARR(13)=1
****        if (iprol.ne.0) goto 20
            call openw(40,0,79,5,19)
      100    CALL SCROLL(0,1,40,0,79,5,19)
      101    CALL INPUTT(0,50,1,'INTERACTION?  $',19,IAR)
           IVAL=iar(1)
           call mcheck('rugoelbatrenstredimoowsp',ival,i)
           if(i.ge.1)goto 300
           call mcheck('RUGOELBATRENSTREDIMOOWSP',IVAL,1)
           if(i.ge.1)goto 300
          CALL TFORM(0,50,2,'INVALID INTERACTION $',19)
          GO TO 101
****  20      i = iprol
      300   CONTINUE
           IF(I.EQ.1) IARR(13)=0
****         IF((I.EQ.1).and.(iprol.eq.0)) CALL closew
****        if(i.eq.1)goto 990
****        if (iprol.eq.2) goto 600
****          IF(I.EQ.2) CALL gotosb
****   605    IF(I.EQ.3) CALL ELEMSB
****         if((i.eq.4).and.(iprol.eq.4)) goto 500
****         if(i.eq.4) call batcsb
       505   IF(I.EQ.5) CALL traced
           IF(I.EQ.6) CALL ESUB
           IF(I.EQ.7) CALL FINISH
           IF(I.EQ.8) CALL REFORM
           IF(I.EQ.9) CALL DISPSB
           IF(I.EQ.10) call monitr
           IF(I.EQ.11) CALL OWNINT
           IF(I.EQ.12) CALL CSPEED
****        if(i.eq.2)i=1
****        if(i.eq.4)i=1
****        if(i.eq.6)i=1
****        if(i.eq.7)i=1
****        if(i.eq.1)goto 300
           GO TO 100

****  500  if (iprol.eq.4) it=ibatim
            if (it.gt.tarr(1)) itemp=it-tarr(1)
            call schedu(-2,itemp,IARR(14))
            IARR(15)=1
            IARR(13)=0
            IARR(6)=0
            goto 505
```

```
****   600  if(iprol.eq.2) it=igotim
            if(it.gt.tarr(1)) itemp=it-tarr(1)
            itime=itemp
            call schedu(-1,itime,iarr(14))
            iarr(15)=1
            iarr(13)=0
            goto 605

990   RETURN
      END
```

Appendix 4

Developing an Intelligent Controller

This appendix presents some of the code that is required to achieve an intelligent controller. The different topics are:

- linking FORTRAN and PROLOG programs : an example;
- passing variable values between model dependent and independent processes;
- developing a non-procedural controller;
- illustration of self-explaining models.

Appendix 4 - a : Linking FORTRAN and PROLOG programs : an example

In this appendix, an example is given on how a FORTAN program may be linked to a PROLOG program. The technical detail is examined concerning the automation of the frame presented in section 3.3.3.

(i) The Fortran component

In section 3.3.3, an experimental frame was automated in a subroutine FRAME1. Part of this frame was:

```
SUBROUTINE FRAME1
      INCLUDE 'CSIM'
      CALL PARAMETERS
      CALL INPUT1('What is the run in time?')
      CALL INPUT1('What is the length of runs?')
      DO I = 1,10
            :
      CONTINUE
RETURN
END
```

These two questions need to be answered from the PROLOG program. To achieve this, a call

must be inserted to the PROLOG program advising it that input is requested regarding run in time and length of runs. This call is implemented in the following way:

```
      SUBROUTINE FRAME1
            INCLUDE 'CSIM'
            INCLUDE 'LSIM'
            CALL INCOMM
            CALL CFILL(1)
            CALL CSEND
            CALL CLRC
            CALL RDI
                  IRUNIN = OINTG
            CALL RDI
                  ILENGTH = OINTG
            DO .....
      RETURN
      END
```

To support this call to the PROLOG are two subroutines which have been described above as being problem dependent.

```
      SUBROUTINE RPRAMS
            INTEGER*1 FLG
            INCLUDE 'LSIM'
            CALL TATOM('frame_1;',FLG)
                  IF (FLG.EQ.1) GOTO 10
10          CONTINUE
      RETURN
      END

      SUBROUTINE CFILL(IMESS)
            INTEGER*2 IMESS
            INCLUDE 'LSIM'
            CALL STORE (IMESS,1,'frame_1;')
      RETURN
      END
```

The second subroutine (CFILL) is used to store the names of all the possible commands executable by the PROLOG. It associates with that command an index number. Here "frame-1" has index 1.

The subroutine RPRAMS controls the input from the FORTRAN to the PROLOG. In this application, only input from PROLOG to FORTRAN is required and hence there is no facility for FORTRAN input.

Looking at the subroutine FRAME_1, there is

```
CALL CFILL(1)
CALL CSEND
```

The first line stores the command which is required.

The second line sends this command through the link.

CALL INCOMM & CALL CLRC are used to clear and prepare the link for commands and parameters to be sent through it.

The lines

```
CALL RDI
variable = OINTG
```

read back integer input from the PROLOG. The integer input is held in OINTG and another variable needs to be instantiated to OINTG before another CALL RDI is made. Here IRUNIN and ILENGTH will hold values for the run in time and length of runs. But how were these values derived ?

(ii) Prolog Program

At the other end of the link, there must be a PROLOG file which contains at least clauses for each of the executable commands stored on the FORTRAN side in the CFILL(IMESS) subroutine. In the example considered, there must be a clause "frame_1". This clause must derive and send back to the FORTRAN controller values for run in times and length of runs. These values in the first instance could be obtained from the user in the following way:

```
frame_1 : -  write('What run in time should be used?')
             read (X), nl
             write('What length of runs are needed?')
             read (Y), nl
             commend,
             writeri(X),
             writeri(Y),!.
```

The clause "commend" indicates that all instructions from the FORTRAN have been received. It prepares the link to send back output from the PROLOG. "writeri(X)" sends an integer value back to FORTRAN. This will be picked up by a CALL RDI in the FORTRAN program.

Appendix 4 - b : Using Variables in Model Dependent and Independent Processes

In this appendix, illustration is given on how different variables that are used in the model dependent and model independent subroutines may be equivalenced to permit the linking of these two types of processes. The FORTRAN implementation used supports the use of "INCLUDE files". These "INCLUDE files" are generally used for the declaration of variables. Common variables are stored in this way. Typically every model has an "INCLUDE file" which appears in every subroutine of the model. In the die-shop example, the INCLUDE file is called DIESIM. The model independent subroutines have "INCLUDE CSIM".

Typically DIESIM looks like

```
integer die(150),
integer iamaq(33), ipmaq(33), ilmaq(33), icarq(4)
integer imo,imtw,imth,ICAR
integer IBD, IDP, IBB
common /entities/ die
common /queues/ iamaq, ipmaq, ilmaq, icarq
common /parameters/ imo, imtw, imth, ICAR
common /performance/ IBD, IDP, IBB
```

and CSIM :

```
integer ibatim, irun
integer irru(10),ipm(10),totv(10),monv(10,10),meav(10),varv(10),
+lobv(10),upbv
integer param(10)

common/general/ibatim, irun
common/monitor/irru,ipm,totv,monv,meav,varv,lobv(10),upbv
common/cpm/ipm
common/cparam/integer param
```

To use subroutines such as PARAMETERS & MONITO it is necessary to equivalence:

IMO with PARAM(1)

IMTW with PARAM(2)

IMTH with PARAM(3)

ICAR with PARAM(4)

IBD with ipm(1)

IDP with ipm(2)

IBB with ipm(3)

Equivalence statements can be set using statements of the form (ibd,ipm(1)).

If two items are equivalenced, only one of them may be held in common. Also these equivalences need only be declared in one or other of the dependent or independent subroutines. To ensure generality of model independent subroutines, it seems appropriate to enforce the following changes on the model's common file.

DIESIM becomes

```
integer die(50)
integer iamaq(33), ipmaq(33), ilmaq(33), icarq
integer ime, imtw, imth, icar
integer IBD, IDP, IBB
integer iseed
integer IPARAM(4)
integer ipm(3)
integer seed(3)
common /entities/dies
common /queues/iamaq, ip,aq, ilmaq, icarq
common /cpm/ipm
common /cparam,param
common /cseed/seed
equivalence (ibd, ipm(1))
equivalence (idp, ipm(2))
equivalence (ibb, ipm(3))
equivalence (ime, iparam(1))
equivalence (imtw, param(2))
equivalence (imth, param(3))
equivalence (icar, param(4))
equivalence (iseed, seed(1))
```

and CSIM remains unchanged.

Appendix 3.c : Development of a non-procedural PROLOG Controller

This appendix presents the necessary code to develop a non-procedural Prolog Controller that activates a Fortran Executioner

(I) The Fortran side

```
      SUBROUTINE EXECUTIONER
            INCLUDE 'LSIM'
1000        CALL CLEAR
            CALL INCOMM
            CALL CFILL(1)
            CALL CSEND
            CALL CLRC
            CALL RD1
            IOPTIO = OINTG
            GOTO (1,2,3,13) IOPTIO

     1CALL FRAME1
            GOTO 1000

     2CALL FRAME2
            GOTO 1000

     3      CALL FRAME3
            GOTO 1000

     13     STOP
            RETURN
            END
```

The EXECUTIONER asks (CFILL(1)) the prolog CONTROLLER to tell it which frame to execute (CALL RDI).

```
      SUBROUTINE FRAME1
            CALL CFILL(2)]
            CALL RD1
            .
            .
            DO 10 I=1,J
10          CONTINUE
            RETURN
      END
```

Each frame asks (CFILL(2)) the Prolog for input to permit execution of the frame (CALL RDI).

Next, the subroutine RPRAMS controls the nature of the interactions that take place between Fortran and Prolog. A "GOTO 10" simply calls the Prolog clause to be executed. The "GOTO 20" means that Prolog requires input from the Fortran. This is used when results are fed back.

```
      SUBROUTINE RPRAMS
            INTEGER*1 FLG
            INCLUDE 'LSIM'
            CALL TATOM('controller;',FLG)
                  IF (FLG.EQ.1) GOTO 10
            CALL TATOM('frame_1;',FLG)
                  IF (FLG.EQ.1) GOTO 10
            CALL TATOM('frame_2;',FLG)
                  IF (FLG.EQ.1) GOTO 10
            CALL TATOM('frame_3;',FLG)
                  IF (FLG.EQ.1) GOTO 10
            CALL TATOM('feed_back;',FLG)
                  IF (FLG.EQ.1) GOTO 20
            GOTO 990
 20         INTG = INTP(1)
            CALL WRRI
            GOTO 990
 10         CONTINUE
        RETURN
        END
```

The subroutine CFILL(IMESS) stores all the available interactions between Fortran and Prolog.

```
      SUBROUTINE CFILL(IMESS)
            INTEGER*2 IMESS
            INCLUDE 'LSIM'
            CALL STORE(IMESS,1,'controller;')
            CALL STORE(IMESS,2,'frame_1;')
            CALL STORE(IMESS,3,'frame_2;')
            CALL STORE(IMESS,4,'frame-3;')
            CALL STORE(IMESS,5,'feed_back;')
```

In this example, frame_3 finds a run in time. This value is sent back using the FEEDBACK process.

```
SUBROUTINE FRAME3
      CALL CFILL(3)
      CALL RD1
        :
      CALL FEEDBACK
      RETRUN
END

SUBROUTINE FEEDBACK
      CALL INCOMM
      INTG(1) = 1
      INTG(2) = RUNINTIME
      CALL CFILL(5)
      CALL CSEND
RETURN
END
```

(ii) The Prolog side

The PROLOG based controller selects the option frame to be executed. It controls the input to these frames as illustrated below:

```
controller :- current_problem(X)
              commend,
              assert(problem(X)),!.


controller :- commend,
              write('Please select:
                    1. - Run an experiment
                    2. - Find run in time
                    3. - Etc
                    4. - Stop '),nl,
              write('Option:'),
              read (Answer),nl,
              asserta(current_problem(Answer)),
              asserta(problem(Answer)),
              select_frame,!.
```

The clauses check that FORTRAN and PROLOG are ready to communicate. The second clause identifies from the user the problem to be investigated (problem(X)). The fact current_problem(X) is stored to remember the number of investigations started. At any one time, there will be only one current_problem(X) under investigation, but there may be several problem(Y). These facts are stored sequentially using the PROLOG 'asserta' clause. This allows retrieval in the input order.

```
select_frame :-   problem(X),
                  X == 1,
                  execute_frame_1,!.

select_frame :-   problem(X),
                  X == 2,
                   |
                  execute_frame_2,!.
```

Once all input has been derived to activate a frame, the EXECUTIONER is told which frame to go to. It will then in turn ask the Prolog for input to this frame. These are found from clauses "frame_(X)".

```
frame_1 :-    irunin(X)
              ilength(Y)
              parameter-1-(A),
              parameter-2-(B),
              parameter-3-(C),
              commend,
              writeri(X),
              writeri(Y),
              writeri(A),
              writeri(B),
              writeri(C),!.

frame_2 :-    condition_1_(X),
              condition_2_(Y),
              parameter_1_(A),
              parameter_2_(B),
              parameter_3_(C),
              commend,
              writeri(X),
              writeri(Y),
              writeri(A),
              writeri(B),
              writeri(C),!.

frame_3 :-

          |
          |
```

Appendix 4 - d : Illustration of self_explaining models

This appendix illustrates the required changes to a model structure in order to turn it into a self-explaining model to be used with an intelligent controller.

The user must change the heading of his program to:

```
      PROGRAM DIESHOP
          CALL FPTALK
          STOP
      END
```

Next the user must declare in the following manner the names of the services queues in his model.

```
      SUBROUTINE QNAMES
          INCLUDE 'DIESIM'
          INCLUDE 'LSIM'
          DO 10 JJ = 1,4
              CALL INCOMM
              IF (JJ.EQ.1) CALL FATM(1,'iamaq(11);')
              IF (JJ.EQ.2) CALL FATM(1,'iamaq(21);')
              IF (JJ.EQ.3) CALL FATM(1,'iamaq(31);')
              IF (JJ.EQ.4) CALL FATM(1,'icarq;')
              CALL CFILL(3)
              CALL CSEND
              CALL CLRC
   10     CONTINUE
        RETURN
        END
```

Declaration of the entities

```
      SUBROUTINES ENAMES
          :
          IF (JJ.EQ.1) CALL FATM(1,'bridge;')
          IF (JJ.EQ.2) CALL FATM(1,'plate;')
          IF (JJ.EQ.3) CALL FATM(1,'back;')
          CALL CFILL(3)
          :
      END
```

Declaration of the parameters

```
      SUBROUTINES PNAMES
            :
            IF (JJ.EQ.1) CALL FATM(1,'imo;')
            IF (JJ.EQ.2) CALL FATM(1,'imtw;')
            IF (JJ.EQ.3) CALL FATM(1,'imth;')
            IF (JJ.EQ.4) CALL FATM(1,'icar;')
            CALL CFILL(3)
            :
      END
```

Declaration of the performance measures

```
      SUBROUTINE PVNAME
            :
            IF (JJ.EQ.1) CALL FATM(1,'ibb;')
            IF (JJ.EQ.2) CALL FATM(1,'ibd;')
            IF (JJ.EQ.3) CALL FATM(1,'idp;')
            :
      END
```

Appropriate changes needed to be made to the link routine RPRAMS and CFILL(imess). On the PROLOG sides, the following clauses were devised to pick up appropriate names:

```
enames :-   reada(Atom),
            commend,
            enamenu(X),
            Number is X
            asserta(ename(Number,[Atom]))
            echeck,!.

qnames :-   reada(Atom),
            commend,
            qnamenu(X),
            Number is X
            asserta(qname(Number,[Atom]))
            qcheck,!.


pnames :-   reada(Atom),
            commend,
            pnamenu(X),
            Number is X
            asserta(pname(Number,[Atom]))
            pcheck,!.
```

```
pvnames :- reada(Atom),
           commend,
           pvnamenu(X),
           Number is X
           asserta(pvnam(Number,[Atom]))
           pvcheck,!.
```

The clauses read the names from the model and associate a "number" fact with the given name. The name can be used when storing results. This name learning process is activated when the model is looking through the subroutine FPTALK which now replaces the old FUTALK.

```
      SUBROUTINE FPTALK
            INCLUDE 'LSIM'
            CALL CLEAR
            DO 5 II=1,20
                  CALL LSNOFF(II)
 5          CONTINUE
            CALL TFORM(5,10,4 Please select how you wish to use model:
                              - as a user controlled simulator
                              - in conjunction with Prolog
                                          controller
            CALL INPUTI(What option please?',I)
            IF(I.EQ.1) CALL logic_of_model
            IF(I.NE.2) GOTO 900
                  CALL INTRO
                  CALL ENAMES
                  CALL QNAMES
                  CALL PNAMES
                  CALL PVNAME
                  CALL EXECUTIONER
900         CALL LSNOFF(5)
            CALL LSNON(1)
            CALL LSNON(2)
      RETURN
      END
```

Appendix 5

WES : Some Rule Operations

In chapter 4, the advisor was described and the principles for addressing some of the experimentation tasks were outlined. The purpose for this appendix is to illustrate the internal operations of WES in making use of the principles proposed. The attention will focus on the use of design rules within a prediction-rule. To this effect, a sample of the default rules of WES are presented in their English translation during the process description.

The PROLOG programming concepts that underlie the structure of WES are the use of pattern matching and the use of production rules. Knowledge may be encoded as facts or as rules. Generally speaking, facts are asserted during the consultation about a given problem whilst the rules exist in the rule-base about appropriate decision-making processes. In this example, a consultation is considered where the following facts have been derived.

- The model is not new to WES : ie WES already has some knowledge about the model and it is linked to a controller (see chapter 5)
- The current problem is a prediction exercise where the user wishes to know about service-queue utilisations, time-in-the-system of entities and some user-defined performance measure.
- The default configuration is to be used.

This investigation joins the operations within WES during the stage where the rule-base is consulted in order to establish an appropriate experiment to conduct. The operations in the rule-base may be viewed as an attempt to satisfy a goal which entails descending through a tree of rules.

At the current stage, WES is attempting to satisfy prediction rule 3. (This means that prediction rules 1 & 2 must have failed).

RULE : pred_rule_(3)

```
---------------------------------------------------------
IF
      (1) must monitor service-queue utilisations
      (2) must monitor time-in-the-system of entities
      (3) must monitor user-defined performance measures
      (4) must use batch runs
      (5) must use experimental frame 1
   &
      (6) can find configuration for next experiment
      (7) can find run-in time using run-in rules
      (8) can find run_length using run_time_rules
      (9) can determine number of runs by using run_num_rules

THEN
      assert experiment to tell user/controller
            adopt frame 1
            use batch runs
            monitor service-queues
            monitor time-in-the-system of entities
            monitor user-defined performance measures
            run in time required
            length of run required
            number of runs required
      update experiment database
---------------------------------------------------------
```

To satisfy this rule a number of facts must be true (conditions 1-5). These will have been asserted during the consultation as facts. WES checks there existence in the rule-base. If this exercise evaluates true (by pattern matching) for all these required facts, operations proceed to condition 6.

Condition 6 : configuration

Determining the configuration that is to be used entails collecting all known facts about the current status of the parameter values. Some may have been changed from their default values during the consultation or by the application of other task rules. This condition should in fact never fail if WES has been provided with correct information.

Condition 7 : run-in rules

A number of rules are available to WES for determining the run-in time to be used.

RULE run_in_rule_(1) :

IF
run-in time has been derived previously
THEN
assert same run-in time

RULE run_in_rule_(2)

IF
supposed to use OWN RULES
THEN
goto OWN RULES for run-in time

RULE run_in_rule_(3)

IF
not linked into a controller
THEN
ask user if he knows suitable run-in time

RULE run_in_rule_(4)

IF
derived from a previous experiment
with same configuration
THEN
assert same run-in time

RULE run_in_rule_(5)

IF
system is known to be terminating
THEN
assert run in time = 0

```
RULE run_in_rule_(6)

        ----------------------------------------------------
        IF
                stand alone version
                user has not suggested suitable run in time
        THEN
                ask user to select method for finding run in time
                from
                        1. Let WES choose
                        2. Decreasing utilisations
                        3. Graph plotting
                        4. No method
        &
        IF
                user does not choose option 4
        THEN
                tell user how apply this technique
                ask user for answer
                assert answer as run in time
        ----------------------------------------------------

RULE run_in_rule_(7)

        ----------------------------------------------------
        IF
                linked to a controller
        THEN
                apply frame to find run in time
        ----------------------------------------------------

RULE run_in_rule_(8)

        ----------------------------------------------------
        IF
                run in time not found
        THEN
                set run-in  time = 1000
        ----------------------------------------------------
```

The order in which rule are input is very important since WES attempts each rule in sequence. In this example, rule 1 will fail because it is assumed that it has not been derived previously (this would be the case if a frame has been fired that had investigated this problem).

Rule 2 fails because this consultation does not make use of extra file of rules.

Rule 3 fails because the system is linked to a controller.

Rule 4 it is assumed succeeds because in a previous consultation, the same configuration was used and a run-in time was derived then.

WES no longer searches to satisfy a run-in time rule and proceeds to the next condition of prediction rule 3.

Condition 8 : length of runs

The following rules are considered from WES rule base.

```
RULE run_time_rule_(1)

    ------------------------------------------------
    IF
        length of run is known
    THEN
        assert length of run
    ------------------------------------------------

RULE run_time_rule_(2)

    ------------------------------------------------
    IF
        supposed to use OWN RULES
    THEN
        goto OWN RULES for run-in time
    ------------------------------------------------

RULE run_time_rule_(3)

    ------------------------------------------------
    IF
        transient behaviour problem
    THEN
        assert length of run = 10000
    ------------------------------------------------

RULE run_time_rule_(4)

    ------------------------------------------------
    IF
        system is not terminating
    THEN
        set length of run equal to run in time
    ------------------------------------------------
```

```
RULE run_time_rule_(5)

     ----------------------------------------------------------
     IF
          it is permissible to ask user
     THEN
          ask user
          assert answer if not 0 as length of run
-----------------------------------------------------------------

RULE run_time_rule_(6)

     ----------------------------------------------------------
     IF
          derived in previous experiment
     THEN
          assert length of run
     ----------------------------------------------------------

RULE run_time_rule_(7)

     ----------------------------------------------------------
     IF
          still not found length of run
     THEN
          assert length of run = 10000
     ----------------------------------------------------------
```

The user is assumed to have not specified a particular length of run and hence rule 1 fails. Rule 2 and rule 3 fail also because no extra rules are considered and it is a prediction exercise. Since the system is not terminating, rule 4 will apply.

Condition 9 : number of runs

The following rules are available to WES.

```
RULE run_num_rule_(1)

     ----------------------------------------------------------
     IF
          number of runs is known
     THEN
          assert number of runs
     ----------------------------------------------------------
```

```
RULE run_num_rule_(2)

    ---------------------------------------------------
    IF
        using sequential runs
        using antithetics
    THEN
        number of runs is 2
    ---------------------------------------------------

RULE run_num_rule_(3)

    ---------------------------------------------------
    IF
        using sequential runs
    THEN
        number of runs is 1
    ---------------------------------------------------

RULE run_num_rule_(4)

    ---------------------------------------------------
    IF
        using batch runs
        using antithetics
    THEN
        number of runs is 14
    ---------------------------------------------------

RULE run_num_rule_(5)

    ---------------------------------------------------
    IF
        using batch runs
    THEN
        number of runs is 12
    ---------------------------------------------------

RULE run_num_rule_(6)

    ---------------------------------------------------
    IF
        using batch runs followed by sequential runs
        using antithetics
    THEN
        number of runs is 20
    ---------------------------------------------------
```

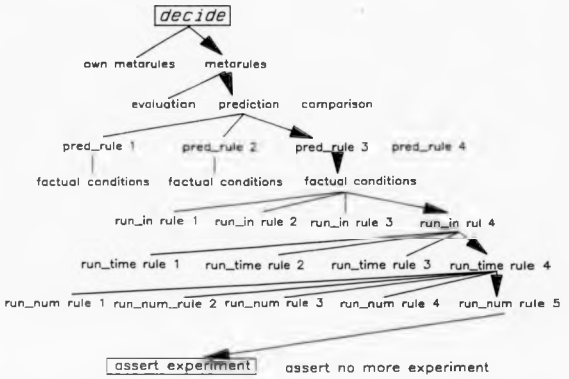
RULE run_num_rule_(7)

IF
using batch runs followed by sequential runs
THEN
number of runs is 14

Condition 4 established that batch runs were to be used but no mention has been made about the use of antithetics. Consequently, the first rule to be satisfied will be rule 5

WES is now in a position to design an experiment to address the problem under consideration. This in turn prompts WES to leave the rule-base and return to the inference engine to execute the next step.

The following diagram illustrates the search pattern through the rule-base. Lines indicate attempted rules. Arrows indicate successful rules.



Appendix 6

Listing of WES output files

Examples of the two different output files that are generated by WES are found in this appendix. These are:

- a model file;
- a report file.

Appendix 6 - a : Listing of a model file generated by WES

In this appendix, part of a file is presented holding all the facts that WES asserts about the model and the experimental results that are asserted during the consultation.

The service queue names and bounds for good and bad average utilisation

```
qname(1,[orders_qu],30,70,1).
qname(2,[cheque_qu],30,70,1).
qname(3,[cash_qu],30,70,1).
qname(4,[credit_qu],30,70,1).
qname(5,[collect_q],30,70,1).
qname(6,[jewel_q],30,70,1).
```

Parameter names with default, lower, upper bound values

```
pname(5,[collect_servers],4,2,5).
pname(4,[credit_servers],3,2,5).
pname(3,[cash_servers],2,2,5).
pname(2,[cheque_servers],1,1,5).
pname(1,[order_servers],2,2,6).
```

Performance measure

```
pvnam(1,[ishop],0,0,0).
```

Seed name for random number streams

```
sname(1,[isearg]).
```

Entity names with lower and upper bounds for average time in the system

```
ename(5,[return_customers],300,500,1).
ename(4,[no_order_customers],300,500,1).
ename(3,[credit_customers],300,500,1).
ename(2,[cheque_customers],300,500,1).
ename(1,[cash_customers],300,500,1).
ename(6,[jewel_customers],300,500,1).
```

Various facts about the model characteristics

```
isubnu(13).
qnamenu(7).
pvnamenu(2).
pnamenu(6).
snamenu(2).
enamenu(7).
bnamenu(1).
```

Aggregate results of queue utilisation with confidence interval

```
qustatus(1,[orders_qu],1,98,96,99,0).
qustatus(2,[cheque_qu],1,9,5,14,0).
qustatus(3,[cash_qu],1,12,8,15,0).
qustatus(4,[credit_qu],1,10,7,12,0).
qustatus(5,[collect_q],1,56,50,61,0).
qustatus(6,[jewel_q],1,41,33,48,0).
```

Aggregate results of entity time in the system with confidence interval

```
enstatus(1,1,3411,3090,3732,0).
enstatus(2,1,3320,3008,3632,0).
enstatus(3,1,3332,2980,3684,0).
enstatus(4,1,287,245,339,0).
enstatus(5,1,8232,7708,8816,0).

enstatus(1,2,991,844,1138,0).
enstatus(2,2,910,774,1046,0).
```

A list of the configurations used in the experiments

```
exconfig(9,2,1,2,2,4,0).
exconfig(8,2,1,2,2,3,0).
exconfig(7,2,1,2,2,4,0).
exconfig(6,2,1,2,4,4,0).
exconfig(5,2,1,3,3,4,0).
exconfig(4,2,2,2,3,4,0).
exconfig(3,3,1,2,3,4,0).
exconfig(2,2,1,2,3,4,0).
exconfig(1,5,3,3,3,3,0).
```

A list of the designs of experiments

```
action(1,8,0,14400,0,10,0,0,0).
action(1,7,0,14400,0,10,0,0,0).
action(1,6,0,14400,0,10,0,0,0).
action(1,5,0,14400,0,10,0,0,0).
action(1,4,0,14400,0,10,0,0,0).
action(1,3,0,14400,0,10,0,0,0).
action(1,2,0,14400,0,10,0,0,0).
action(1,1,700,700,0,6,1,0,0).
```

A record of the number of experiments performed

```
expnumber(9).
```

A list of the WES parameter values used

```
wesp(2,[n]).
wesp(3,[n]).
wesp(4,[3]).
wesp(5,[y]).
wesp(6,[y]).
wesp(7,[n]).
wesp(8,[y]).
wesp(1,[10]).
wesrip(1,[10],10).
wesrip(2,[y],1).
wesrip(3,[y],1).
wesrip(4,[y],1).
wesrip(5,[y],1).
```

A list of the average total queue utilisation and total entity time in the system

```
wfac(904,8,2778).
wfac(904,7,2719).
wfac(904,6,2712).
wfac(904,5,2703).
wfac(904,4,2675).
wfac(904,3,1534).
wfac(904,2,2713).
wfac(904,A,100).
wfac(903,1,29).
```

A recording of all the individual results for each run of every experiment

```
wfac(900,1,[cash_customers],1,592).
wfac(900,1,[cheque_customers],1,666).
wfac(900,1,[credit_customers],1,685).
wfac(900,1,[no_order_customers],1,291).
wfac(900,1,[return_customers],1,373).
wfac(900,1,[jewel_customers],1,498).
wfac(900,1,[orders_qu],1,99).
wfac(900,1,[cheque_qu],1,30).
wfac(900,1,[cash_qu],1,10).
wfac(900,1,[credit_qu],1,11).
wfac(900,1,[collect_q],1,73).
wfac(900,1,[jewel_q],1,100).
wfac(900,1,[ishop],1,-2).

wfac(900,1,[cash_customers],2,484).
```

A recording of all the evaluation of the different configurations

```
wfac(200,8,2778,['Best performance']).
wfac(100,2,2713,['This model performs very well.  Score : ']).
wfac(100,3,1534,['This model performs very well.  Score : ']).
wfac(100,4,2675,['This model performs very well.  Score : ']).
wfac(100,5,2703,['This model performs very well.  Score : ']).
wfac(100,6,2712,['This model performs very well.  Score : ']).
wfac(100,7,2719,['This model performs very well.  Score : ']).
wfac(100,8,2778,['This model performs very well.  Score : ']).
```

Appendix 6 - b : Example report file

In this appendix, an example report file from a WES consultation is presented.

REPORT FILE FOR MODEL : argon

Model : argos
File : argos.rpt

Record of consultation held
by : rick on : 14

During this consultation, WES was asked to optimise the model.

Performance was assessed using : WES default evaluation mechanism (absolute).

The best performance achieved was 65 using configuration :
order_servers = 4
cheque_servers = 2
cash_servers = 2
credit_servers = 2
collect_servers = 3

Ex numb	Perf	order_servers	cheque_servers	cash_servers	credit_servers	collect_servers
1	49	2	1	2	3	4
2	61	3	1	2	3	4	1
3	66	4	1	2	3	4	1
4	63	5	1	2	3	4	1
5	67	4	2	2	3	4	2
6	65	4	3	2	3	4	2
7	51	4	2	3	3	4	3
8	65	4	2	2	4	4	4
9	68	4	2	2	2	4	4
10	67	4	2	2	2	5	5
11	69	4	2	2	2	3	5
12	69	4	2	2	2	2	5
13	66	5	2	2	2	3	1
14	67	4	3	2	2	3	2
15	57	4	2	3	2	3	3

Appendix 7

Implementations for linking Controller-Advisor

The appendix presents different parts of code to illustrate:

- integration of controller and advisor using RS232 parallel link;
- batch-file programming code for sequential link;
- illustration of communication protocol between Prolog and Fortran;
- required changes to existing MICROVISION models for use with WES.

Appendix 7- a : Integrating controller and advisor using parallel link

In chapter 5, it is argued that the advisor and the controller share the same expert model and hence used the same experimental frames. In consequence, the advisor holds the input for the controller. Linking advisor and controller is presented here in a technical overview.

It was seen how the design and execution phase of experimentation is catered for by the subroutine SOLVE.

```
SUBROUTINE SOLVE
 CALL DECIDE
 CALL CONTROLLER
 CALL DISCUS
RETURN
END
```

The reasoning process is carried out by the DECIDE step and the CONTROLLER routine executes the experimentation. From the intelligent controller point of view, these processes may be seen as the specification and selection of the frame to be executed and can be presented as a modification of the subroutine EXECUTIONER of chapter 3.

```
      SUBROUTINE CONTROLLER
       INCLUDE 'LSIM'
1000   CALL CLEAR
       CALL INCOMM                    ; Receive frame selection from
       CALL CFILL(1)                  ; Advisor rather than user
       CALL CSEND
       CALL RDI
       IOPTIO = OINTG
       GOTO(1,2,3,13)IOPTIO

1      CALL FRAME1
       CALL RESULTS
         .
         .
         .
      RETURN
      END
```

(a) Setting up the experiment

The subroutine FRAME1 would be modified to trigger off the clause 'frame_1' presented in chapter 4.

```
      SUBROUTINE FRAME1
        .
        .
       INCLUDE 'LSIM'
       CALL INCOMM
       CALL CFILL(2)                  ; Receive input
       CALL CSEND
       CALL CLRC
       CALL RDI
        IRUN = OINTG
       CALL RDI
        IGOTIM = OINTG
          .
          .
          .
        DO 10 IJ = 1,IRUN
          CALL SLOGIC                 ; Run model
10      CONTINUE
      RETURN
      END
```

The input values are obtained from the adapted clause "frame_1" :

```
frame_1:- commend,
        expnumber(Hh),
        action(1,E,Irutim,Ibatim,Ilastrun,Irun,Prnqu,Prnqo,Ppmpv)
        writeri(Irun),
       writeri(Ibatim),
            .
            .
            .
        writeri(Prnpv),!.
```

Having received all required input to the frame, the controller then executes the frame by automatically running the model as before.

(b) Reading the Results

Thereafter the subroutine RESULTS controls the return of all the results to the PROLOG advisor. Again, a clause "results" is associated with the subroutine RESULTS.

```
      SUBROUTINE RESULTS
       CALL RESBQU
       CALL RESBQO
       CALL RESBPV
      RETURN
      END

      SUBROUTINE RESBQU
       INCLUDE 'ESIM'
       DO 10 JJ = 1,IQNAMENU
        DO 20 JK = 1,IRUN
         CALL INCOMM
         INTP(1) = MONV(JJ,JK)          : send results of utilisation
         INTP(2) = JK                   : of queue JJ during run JK
         INTP(3) = JJ
         CALL CSEND
         CALL CLRC
20      CONTINUE
10     CONTINUE
      RETURN
      END
```

Associated with CFILL(11) is :

```
CALL STORE(IMESS,11,'resbqu;')
```

which will cause the clause 'resbqu' to stand ready to receive the utilisation of the specified queue.

```
resbqu :- readi(I),
          readi(J),
          readi(K),
          expnumber(Hh),
          A is Hh-1,
          assert(qustatus(I,Hh,J,K)),!.
```

The other results are read back in a similar fashion using similar subroutines and clauses.

Appendix 7- b : Implementation of the Sequential Link

In this appendix, the batch file control programs are provided for the sequential link. They are taken from a technical note on the use of WES

Linking Prolog WES to Fortran Simulation

It is possible to link WES directly to the simulation model under consideration.

The execution of WES is prompted from the default drive by the command

C>model filename

This command calls the batch file 'model.bat' with the following contents :

wesarityAr3 %1

The result is to call the batch file Ar3.bat on the drive (wesarity) where the arity programs are loaded. The batch file will take one parameter. This will be the parameter 'filename' that was originally called by model.bat. 'Filename' is the name of the simulation model under consideration.

The contents of 'ar3.bat' are given below :

```
1    echo off
2    set editor=Edit.exe
3    path Wesarity
4    set filepath=Wesarity
5    set comspec=Command.com
6    del prolog.ini
7    copy stwes8 stwes
8    copy mowes8 mowes

9    ren stwes prolog.ini
10   c:WesarityApi
11   ren prolog.ini stwes
12   if exist c:Newprologtop  goto abc
13   %1

14   ren mowes prolog.ini

15   :loop
16   c:WesarityApi
17   if exist c:Newprologtop  goto abc
18   %1
19   goto loop

20   :abc
21   ren prolog.ini mowes
22   del c:Newprologtop
23   set editor=
24   set filepath=
25   path c:;
```

Lines 1 to 8 : sets the arity path up.

Lines 9 to 13 : starts a consultation. First the file 'stwes' becomes the Prolog initialisation file. This is like an autoexec file in MSDOS. It will call up WES (line 10) when the arity program is called. When WES is left a check will be made for the existence of a temporary file called 'ssstop'. This file is created if the consultation with WES has finished. If the file exists the execution of operations jumps to label abc (line 20). If the file does not exist, then this will mean that WES wishes for the simulation model to be run. (line 13).

Line 14 : changes the Prolog initialisation file. The file 'mowes' is used to continue a consultation.

Lines 15 to 19 : deals with the loop. WES is called and consulted. When WES is left a check is made for the existence of a temporary file called 'ssstop'. If the file exists the execution jumps from the loop and goes to label 'abc' (line 20). If the file does not exist, this will

mean that the simulation model is to be executed (line 18). After execution, operations return to the top of the loop and into WES.

Lines 20 to 25 : prepares to quit. The temporary file 'ssstop' is deleted. The path is reset.

Appendix 7 - c : Communications protocol between Advisor and Controller

These notes are taken from technical note 4 on the use of WES.

Communications between Prolog WES and Fortran Simulation

Prolog WES and Fortran Simulation model communicate via shared input and output files. These files are 'HI' , 'BYE' and 'RTIME'.

Prolog WES always talks first. It does so when it wishes the simulation model to run. It opens a file called 'HI'. In the following order, it inputs :

1 iext (values 0 or 1 : indicates type of experiment)
2 iprol (values 1,2,4 : indicates mode of runs : batch,normal)
3 wesp(3) (range acceptance for run in times)
4 tbexp(R) (number of subroutines in the model)
5 irutim (run in time)
6 ibatim (length of run)
7 pnamenu (number of parameters)
8 qnamenu (number of queues)
9 pvnamenu (number of performance measures)
10 snamenu (number of seeds)
11 values for the parameters
12 values for the seeds

The Fortran side will open the file 'HI' and use it as input. After execution it opens the file 'BYE' and outputs :

1 The average utilisations of all the queues
2 The values of the performance measures.

Prolog WES will resume consultation by using this file as input.

Special cases :

The file 'RTIME' is used to collect the time for the lenght of runs. This happens when WES is trying to determine the appropriate run in time for the experiment under consideration. Also input in this file are the average utilisations of the whole system. easured every 100 time units.

Appendix 7 - d : Required changes for Microvision models for use with WES

The documentation presented below is taken from the second technical note on the use of WES.

Adjusting Simulation Model for use with WES

A number of small changes are necessary in order to use a simulation model with WES : a couple of inclusions and four extra subroutines.

1. In the main program of the simulation, the following call must be added :

```
CALL SIMWES
```

Subroutine SIMWES will set up WES for use with this model.

2. In the GOTO(1,2,3,4,....,X)IEVENT line, add X+1 :

```
1000   CALL ADVANC(IEVENT,ITIME,IELE)
       GOTO(1,2,3,4,....,X,X+1)IEVENT

1      CALL SUBX
       GOTO 1000

2      CALL SUBY
       GOTO 1000

       .
       .
       .

X+1    CALL WESSUB(IELE)
       goto 1000
```

3. Parameters : Decide what are the parameters of the model that can be varied. In the subroutine below they must be set to the value of IWES(JJ). Add this subroutine to your program

:

```
      SUBROUTINE PARWES
       INCLUDE 'WESSIM'
       INCLUDE 'name of the common file of model'

either    call setatt(arates,1,IWES(1)),
          call setatt(arates,2,IWES(2)),

or        parameter3 = IWES(3)
          parameter4 = IWES(4)

      RETURN
      END
```

4. Seeds : it is necessary to ensure that randomly generated numbers have a variable seed. This is required to replicate experiments. The variable seed should be set to the value of SWES(JJ). Add this subroutine to your program :

```
      SUBROUTINE SEEWES
       INCLUDE 'WESSIM'
       INCLUDE 'name of common file of model'
       iseed1 = SWES(1)
       iseed2 = SWES(2)
      RETURN
      END
```

5. Performance measures : any measure of performance that you wish to consider should be equated to IPWES(JJ). Include this subroutine in your program.

```
      SUBROUTINE RESWES
       INCLUDE 'WESSIM'
       INCLUDE 'name of common file of model'
       IPWES(1) = perfmeasure1
       IPWES(2) = perfmeasure2
      RETURN
      END
```

6. Queue utilisations : the size of the queues that are to be considered must be set equal to IQWES(JJ). Add this subroutine to your program.

```
      SUBROUTINE QUTWES
       INCLUDE 'WESSIM'
       INCLUDE 'name of common file of model'
       IQWES(1) = ISIZE(queue1)
       IQWES(2) = ISIZE(queue2)
       IQWES(3) = ISIZE(queue3)
       IQWES(4) = ISIZE(queue4)
      RETURN
      END
```

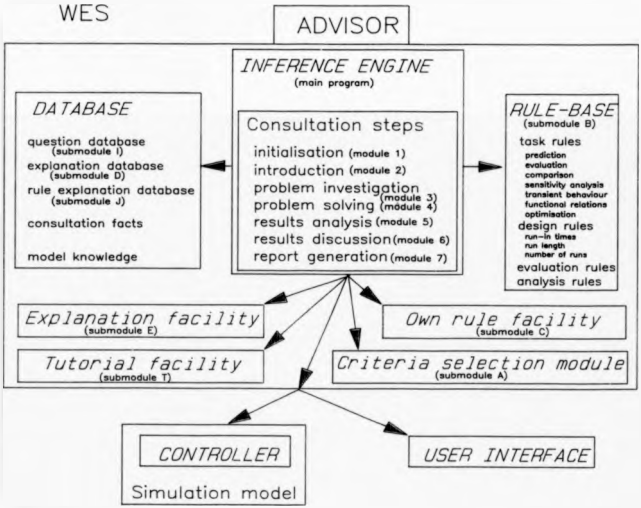
7. Change the linker file :

```
modelname,
n
n
n
n
c:modelname,C:WES,c:sys0,C:SYSUT,c:sys.....
```

Appendix 8

Structure of WES

This appendix provides a summary, in diagram form, of the structure of WES.



WES is seen to be made up of two main components: the Advisor and the Controller. The Advisor's main program directs operations. It retrieves information from the simulation model via the Controller or from the user via a user interface.

To assist the inference engine in consultation operations, there is data knowledge which is situated in the database and there are rules to be manipulated in the rule-base.

There is system knowledge in the database such as the question database, the explanation database and the rule explanation database. There is model knowledge which stores the understanding of WES of the model that is being investigated. And there is consultation knowledge which are facts that WES asserts about the consultation as it proceeds.

The rule-base contains rules to address the different tasks, rules to design experiments, rules to analyse results and rules to evaluate them.

WES also supports modules for an explanation facility, a tutorial facility, an OWN rule facility as well as a criterion selection module.

Appendix 9

Results of Retroactive and MBA studies

In this appendix the results of experiments described in chapter 7 are presented concerning:

- retroactive studies of existing experiments
- comparison of system performance in relation to MBA results.

Appendix 9 - a : Results of retroactive studies

Details of the retroactive studies are presented in this appendix. First, there is the detailed classification of student projects.

M	P	C	R	D	L	R	A	Criteria	Frame	Analysis
airfield	C	3	n	n	650	12	n	T in S	M	Ho
petrol	C	4	n	n	480	7	n	Own	M	Ho
haidreser	C	2	n	n	480	8	y	T in S	M	C I
post	E	4	n	n	1000	1	n	T in S	L (4)	C I
airport	F-R	6	y	n	400	4	n	T in S	M	C I
refuse	S-A	5	y	n	10000	4	n	T in S	M	C I
bank	P	3	y	n	2000	5	y	T in S	M	C I
cafe	P	3	y	n	10100	5	y	T in S	L	C I
laundrette	C	3	n	n	660	5	y	C F	M	Ho
fast food	C	6	n	n	10000	10	n	T in S	M	C I
immigration	O	17	n	n	3500	20	y	T in S	M	C I
hair salon	C	5	n	n	1050	10	n	T in S	M	C I
warehouse	C	5	n	n	540	20	n	T in S	M	C I
petrol	E	-	-	-	-	-	-	-	-	-
food	P	1	y	n	15000	6	y	T in S	M	C I
post office	C	5	n	n	3600	20	n	T in S	M	Ho
shop	C	3	n	n	10000	2	y	T in S	L (5)	C I
registration	O	21	n	n	390	6	y	T in S	M	C I
sports centre	P	6	n	n	1080	4	n	T in S	L (5)	C I
shop	E	1	y	n	1200	1	n	Dynamic changes		
laundrette	O	12	n	n	8000	1	n	T in S	M	C I
post office	P	4	n	n	12000	4	y	T in S	L	C I
crops	S-A	16	y	n	200	5	n	Own	M	C I
harbour docks	E	-	-	-	-	-	-	-	-	-
Abbey National	C	2	y	n	1080	6	-	T in S	M	C I
take away	C	2	y	n	115	10	y	T in S	M	C I
registration	C	5	y	n	180	10	y	T in S	M	C I
garage station	C	2	y	n	40000	5	n	T in S	M	C I
cycle shop	S-A	6	y	y	10000	6	y	own	M	C I
service station	P	3	n	n	20000	1	n	T in S	M	C I
snackbar	C	5	y	n	10000	10	y	T in S	M	C I
garage station	O	9	y	y	720	10	y	T in S	M	C I
manufacturing	S-A	12	y	n	5000	1	n	Q-O	M	C I
food	S-A	4	n	n	1800	1	n	T in S	M	C I
bank	C	4	n	n	1500	1	n	Q-O	M	C I
tesco	C	5	y	n	10000	10	y	T in S	M/L	Ho
stall	C	3	y	n	106000	1	y	T in S	L (10)	C I
cinema	C	5	n	n	300	1	n	T in S	M	C I
petrol	E	-	-	-	-	-	-	-	-	-
supermarket	C	5	n	n	5760	11	y	T in S	M	Ho
milk dairy	C	8	y	n	700	20	y	T in S	M	C I
post office	E	-	-	-	-	-	-	-	-	-
post office	E	-	-	-	-	-	-	-	-	-
library	E	1	y	n	40000	1	-	T in S	M	-
petrol station	E	1	-	-	300	1	-	T in S	M	-
photo lab	C	3	y	n	10000	1	-	T in S	L (5)	C I
registration	O	5	n	n	3600	10	n	T in S	M	C I
service station	S-A	3	n	n	300	18	n	T in S	M	C I
clinic	E	1	n	n	1000	n	n	T in S	M	C I
airport	E	1	n	n	32700	n	n	T in S	M	C I
emergency serice	P	3	n	n	5610	12	n	own	M/L	C I
BBC	C	3	n	n	600	7	y	T in S	M	Ho
ski resort	E	-	-	-	-	-	-	-	-	-
Traffic lights	P	4	y	n	1100	11	n	T in S	L	C I
petrol station	E	-	-	-	-	-	-	-	-	-
post office	E	-	-	-	-	-	-	-	-	-
petrol station	T-B	9	n	n	720	10	y	T in S	M	C I

where :

M = Model problem
T = Task (E = Evaluation, O = Optimisation, P = Prediction, C = Comparison,
S-A = Sensitivity analysis, F-R = functional relation,
T-B = transient behaviour.
C = number of different configurations considered
R = Awarness of run-in issues (y = yes , n = no)
D = Awareness of need for different run in according to configuration
L = Length of runs
R = Number of replications
A = Use of antithetics
Criteria = evaluation criteria
Frame = experimental frame (M = multiple runs with single set of observations
L = long runs with mutiple set of observations).
Analysis = test applied to results.

0-1:	24%
0-2:	31%
0-3:	50%
0-4:	61%
0-5:	79%
0-6:	86%
0-8:	88%
0-9:	91%
0-15:	95%
0-20:	98%
0- :	100%

Table a7.1 : Cumulative percentage of number of configurations considered

Time in the system	: 72%
Own	: 7%
Queue Occupancy	: 3%
C F	: 1%
Unknown	: 15%

Table a7.2 : Percentage use of different criteria

Multiple short runs	: 70%
Long runs	: 17%
Unknown	: 16%

Table a7.3 : Percentage use of experimental frames

Hypothesis testing	: 12%
Confidence intervals	: 68%
None	: 19%

Table a7.4 : Percentage use of formal analysis

Yes : 37%
No : 47%

Unknown : 16%

Table a7.5 : Percentage awareness of run in time considerations

---

Yes : 10%
No : 90%

Table a7.6 : Percentage awareness of different run in time considerations

---

1	**********	25%
2		
3		
4	**	
5	****	31%
6	****	
7	*	
8	*	
9		
>=10	**********	44%
>=20	****	

Table a7.7 : Number of replications for multiple runs

Yes : 35%
No : 44%

Unknown : 21%

Table a7.8 : Percentage use of antithetics

Appendix 9 - b : Results of MBA studies

In this appendix, the details of the WES versus MBA experiment are presented.

Name	Configuration Or	Ch	Ca	Cr	Co	S	A	Rin	L	R	Criteria	Frame	Analysis
chil	5	3	2	2	4	16	2	n	39600	20	Own Cost	M	CI
alle	5	3	2	2	3	15	11	n	7200	12	V & T	M	M
carr	4	3	2	2	4	15	9	y	28800	30	T	M	CI
thor	4	2	2	2	3	13	5	n	15000	15	T	M	CI
roke	5	3	2	2	4	16	7	n	14400	16	V	M	M
lawe	5	3	2	3	4	17	5	n	1000	15	V & T	M	CI
andr	5	3	2	2	4	16	6	y	100000	1	T	L(20)	CI
warn	5	3	2	2	4	16	6	n	14400	15	C & V * T	M	CI
wagl	5	3	2	2	4	16	3	n	14400	18	V & T	M	CI
halm	5	3	2	2	3	15	6	y	28800	15	V & T	L(15)	CI
mall	5	3	2	2	3	15	3	n	14400	9	V & T	M	CI
smit	5	3	2	2	4	16	-	n	14400	15	T & C	M	CI
diaz	5	3	3	3	5	19	7	y	111600	1	V & C	L(30)	CI
suza	6	3	4	3	4	20	3	y	39600	1	T & C	L(10)	M
dick	5	3	2	3	4	17	6	y	30600	1	T	L(8)	CI
chan	5	3	2	2	4	16	8	y	128000	1	T	L(30)	CI
chau	6	3	2	2	4	17	7	y	111600	1	T & C	L(30)	CI
ngaa	5	3	2	2	4	16	6	n	14400	30	T & C	M	CI
liaa	6	3	2	2	4	17	9	y	111600	30	T & C	L(30)	CI
jess	5	3	2	3	4	16	8	n	14400	16	T	M	CI
boga	5	3	2	3	4	17	15	y	14400	30	V	M/L(30)	M
well	5	3	3	3	4	16	8	y	15120	1	T & V	L(16)	CI
yuen	6	3	3	2	4	15	7	y	28800	1	V & T	L(15)	CI
chab	5	3	2	2	4	16	8	y	111600	1	T & C	L(30)	CI
ausa	5	3	2	2	4	16	7	y	180000	1	T	L(35)	CI
rice	5	3	2	2	4	16	3	y	6000	10	T & C	M	CI
bret	5	3	2	2	4	16	6	y	55800	1	T & C	L(30)	CI
hutc	4	3	2	2	4	15	2	n	14400	15	T	M	CI
chua	5	3	2	2	4	16	8	n	1800	15	T	M	CI
stoc	5	3	3	2	4	17	7	n	216000	1	T & C	L(30)	CI
skid	4	3	2	2	4	15	7	n	14400	6	T & C	M	CI
lama	5	3	3	3	4	18	7	y	118800	1	T & C	L(30)	M
toda	5	3	2	2	3	15	6	n	14400	1	T	M	CI
lawa	5	3	2	2	4	16	9	n	14400	1	T & C	M	CI
hung	5	3	2	2	4	16	7	n	14400	30	T	M	CI
gree	4	3	2	2	4	15	10	y	14400	1	T	L(9)	CI
hoaa	4	3	2	2	4	15	10	y	14400	1	T	L(9)	CI
perr	5	3	2	3	4	17	6	y	57600	1	T	L(30)	CI
wong	5	3	2	2	3	15	-	y	7200	15	T & C	M	CI

where:

- Name is project reference
- Configuration denotes MBA recommendations for staffing
  - Or number of staff on order desk
  - Ch number of staff on cheque counter
  - Ca number of staff on cash counter
  - Cr number of staff on credit counter
  - Co number of staff on collect counter
- S denotes the total number of staff proposed
- A is number of alternatives examined
- Rin tells whether run-in time was considered
- L denotes the length of runs used
- R is number of sets of sampled taken per experiment
- Criteria denotes the criteria used for evaluation
  - Own Cost means user defined
  - V means use of visual observation
  - T means time-in-the-system
- Frame records how samples were collected
  - M = multiple short runs
  - L (X) = X number of long runs
- Analysis denotes the model of analysis used
  - CI = confidence intervals
  - M = mean values only

Table a8-1 : Analysis of MBA results

```
 2 : * *
3 : * * * *
4 :
5 : * *
6 : * * * * * * * *
7 : * * * * * * * * *
8 : * * * * *
9 : * * *
10 : * *
11 : *
12 :
13 :
14 :
15 : *
```

Table a8-2 : Distribution of the number of alternatives in MBA experiments

```
=< 4  : * *
=< 9  : * * * * *
=< 14 : * * *
=< 19 : * * * * * * * * * * * * * *
=< 24 : * *
=< 29 :
=< 34 : * * * * * * * * * * * *
>34   : *
```

Table a8-3 : Distribution of sample sizes for MBA experiments

```
Confidence Intervals : 34
Means :  5
```

Table a8-4 : form of analysis for MBA results

```
Long runs with multiple sets of observations :  48%
Short runs with single set of observations : 52%
```

Table a8-5 : Percentage usage of frames

Time in the system : 90%
Cost Function : 41%
Visual observation : 28%

Table a8-6 : Percentage Use of different criteria

Appendix 10

Linking WES to Data-driven Simulations

In this appendix, illustration of some of the components of the work that involved linking WES to WITNESS, data-driven simulation environment. The following are presented:

- an example data-file of a WITNESS simulation model;
- an example report file produced by WITNESS;
- an example "instruction file" that WES is taught to produce;
- the batch file program for WES-WITNESS integration;

Appendix 10 -a : Example data-file of WITNESS model

In this appendix, part of a data file is given of a WITNESS simulation model.

```
! WITNESS MODEL: ARGON

  * Title: ARGON STORE
  * Author: RDH
  * Date: 1/3/88

DEFINE

  Part: CASH,Variable attributes;
  Part: CHEQ,Variable attributes;
  Part: CRED,Variable attributes;
  Part: VIEW,Variable attributes;
  Part: RETN,Variable attributes;
  Part: JEWL,Variable attributes;
  Buffer: ORDERQ,1,1000;
  Buffer: CASHQ,1,1000;
  Buffer: CHEQQ,1,1000;
  Buffer: CREDQ,1,1000;
  Buffer: COLLQ,1,1000;
  Machine: ORDER,5,Single,0,0;
  Machine: CASHPY,3,Single,0,0;
  Machine: CHEQPY,3,Single,0,0;
  Machine: CREDPY,3,Single,0,0;
  Machine: COLLPY,3,Single,0,0;
  VAriable: STIME,1,Real;
  Attribute: TYPE,1,Integer,1;

END DEFINE

DISPLAY

  OPTIONS

    Mouse: NO
    Background display
      * window: 1,0
      * window: 2,0
      * window: 3,0
      * window: 4,0;
    Grid: YES

  ! DEFAULTS
```

Appendix 10 - b : Example report file generated by WITNESS

Page 1 ARGON REPORT Time 1440.00

PART	ENTERED	SHIPPED	SCRAP	ASSEMBD	REJECT	WIP	AV WIP	AV TIME
CASH	10	4	0	0	0	6	3.14	452.09
CHEQ	7	2	0	0	0	5	1.77	363.95
CRED	5	2	0	0	0	3	1.87	537.67
VIEW	6	4	0	0	0	2	0.47	112.71
RETN	5	5	0	0	0	0	0.25	72.10
JEWL	3	3	0	0	0	0	0.23	112.65

Page 2 ARGON REPORT Time 1440.00

MACHINE	NO OPS	%IDLE	%BUSY	%BLOCK	%SETUP	%DOWN	%WAIT
ORDER(1)	5	48.97	51.03	0.00	0.00	0.00	0.00
ORDER(2)	4	32.46	67.54	0.00	0.00	0.00	0.00
ORDER(3)	2	24.17	75.83	0.00	0.00	0.00	0.00
ORDER(4)	3	38.54	61.46	0.00	0.00	0.00	0.00
ORDER(5)	3	36.38	63.62	0.00	0.00	0.00	0.00
CASHPY(1)	2	50.09	49.91	0.00	0.00	0.00	0.00
CASHPY(2)	1	43.76	56.24	0.00	0.00	0.00	0.00
CASHPY(3)	1	46.78	53.22	0.00	0.00	0.00	0.00
CHEQPY(1)	2	41.63	58.37	0.00	0.00	0.00	0.00
CHEQPY(2)	2	80.96	19.04	0.00	0.00	0.00	0.00
CREDPY(1)	0	16.22	83.78	0.00	0.00	0.00	0.00
CREDPY(2)	2	72.69	27.31	0.00	0.00	0.00	0.00
COLLPY(1)	5	80.82	19.18	0.00	0.00	0.00	0.00
COLLPY(2)	4	77.26	22.74	0.00	0.00	0.00	0.00
COLLPY(3)	4	81.35	18.65	0.00	0.00	0.00	0.00
COLLPY(4)	3	70.96	29.04	0.00	0.00	0.00	0.00

Page 3 ARGON REPORT Time 1440.00

BUFFER	TOTAL IN	OUT	NOW	MAX	MIN	AV SIZE	AV TIME
ORDERQ	28	24	4	4	0	0.07	3.65
CASHQ	7	7	0	1	0	0.04	8.57
CHEQQ	5	5	0	1	0	0.00	0.00
CREDQ	3	3	0	1	0	0.05	23.03
COLLQ	18	18	0	1	0	0.00	0.00

Page 4 ARGON REPORT Time 1440.00

VARIABLE	VALUES
STIME	90.1512

Appendix 10 -c : Example "instruction file"

This appendix presents a typical instruction file that WE'S prepares to conduct experiments.

```
NO,ARGON
GO
SELECT,ORDER,Q,5;
SELECT,CASHPY,Q,3;
SELECT,CHEQPY,Q,2;
SELECT,CREDPY,Q,2;
SELECT,COLLPY,Q,4;
BATCH,DURATION,1440
REPORTS
OUTPUT,FILE,ARGON
;
KILL
;
```

This file sets all the parameter values by using the SELECT command by specifying (Q) the quantity eg 5 for parameter ORDER. Next it specifies the length of the run (BATCH,DURATION,1440) and asks for output in file ARGON. Finally, it arranges to exit from the simulation environment (KILL).

Appendix 10 -d : Modification of Sequential link for WITNESS

In this appendix, the modified link batch file and modified WITNESS command file are united to give the following WES/WITNESS link.

```
REM ********* Set environment up ************

echo off
set editor=\dit.exe
path
196set filepath=
196set comspec=del prolog.ini
copy stwes8 stwes
copy mowes8 mowes

REM ********** Start WES consultation ************

ren stwes prolog.ini
c
265ren prolog.ini stwes
ren mowes prolog.ini

if exist 0wprolotop goto abc

:loop
REM ***************************************

REM START OF WITNESS

REM ECHO OFF
ECHO [3A[K[B[K[B[K[3A
IF NOT EXIST XXCFI.TXT GOTO CONT1
DEL XXCFI.TXT
:CONT1
HOME SAVE
IF NOT EXIST ASSIGN.SYS GOTO CPYGEM
IF NOT EXIST IBMEHFP6.SYS GOTO CPYGEM
IF NOT EXIST GEMVDI.EXE GOTO CPYGEM
GOTO BEGIN
:CPYGEM
COPY GEMSYSIBMEHFP6.SYS . >NUL
COPY GEMSYSASSIGN.SYS . >NUL
COPY GEMSYSGEMVDI.EXE . >NUL
:BEGIN
```

```
IF EXIST %1 GOTO RUN1
GOTO RUN
:RUN1
COPY %1 XXCFI.TXT >NUL
:RUN
COMMAND /C WITIN
GEMVDI /WIT.EXE >NUL:
HOME GO
COMMAND /C WITOUT
IF EXIST XXCFI.TXT DEL XXCFI.TXT

REM END OF WITNESS

:EXIT

REM ********** Return to WES consultation **********

command /c:
265if exist c:0wprolotop goto abc

REM ********** Loop to return to WITNESS if required **********

goto loop

:abc

REM ********** Leave WES consultation **********

ren prolog.ini mowes
del c:0wprolotop
set editor=
set filepath=
cls
type 196e
path c:;
```

Appendix 11

Files of extra EXPERTISE for WES

This appendix contains two files that may be consulted during consultation time to replace the default rules of WES. The different files are:

- WITEXP - expertise for use with WITNESS models;
- EXPERT - expertise for Optimisation problem using uncertainty factors.

Appendix 11 - a : File for use with WITNESS models

```
/*************************************************************************/
/*************************************************************************/
/**********
/**********      Expertise for USE with WITNESS Package
/**********
/*************************************************************************/
/*************************************************************************/

/*  Rules to check for steady state

wi(wriq(A)) :-
        write('Checking steady state... '),
        A > 3,
        Na is A, Nb is A-1, Nc is A-2, Nd is A-3,
        wfac(722,Na,Ea),
        wfac(722,Nb,Eb),
        wfac(722,Nc,Ec),
        wfac(722,Nd,Ed),
        not(Ea == 0),
        not(Eb == 0),
        not(Ec == 0),
       write('testing'),

       ((Ec == Eb,Ed < Ec);(Ec =< Eb)),
       Ea < Eb,
        not(Ed == 0),
        wesrip(7,[X],Y),
        Aa is A*Y,
        write('Run in time has been found : '), write(Aa),nl,
        assert(exrutim(Aa)),!.

wi(wriq(A)) :-          wesrip(6,[X],Y),          A < Y,!.
```

```
wi(wriq(A)) :-
        expnumber(R),
        assert(exrutim(3000)),
        assert(wfac(100,R,-99,['Unstable configuration '])),!.

wi(run_in_(1)) :-
        prleg(5),
        retract(iext(1)),
        asserta(iext(0)),
        asserta(comtry(1)),
        assert(action(1,0,0,0,0,0,0,0,0)),
        write('run in rule 8 applies'),nl,
        runten,!.


/* Rules to determine suitable frame

pred_rule_(0) :-
        prleg(5),
        not(frame(X)),cls,
        choice_frame,
       nl,nl,write('>>>>>> '),
        question(860),
        answer(860,[A]),nl,nl,
        wesp(9,[U]),
        (not(A == 1);(A == 1,asserta(frame(1)))),
        (not(A == 2);(A == 2,asserta(frame(2)))),
        (not(A == 3);(A == 3,asserta(frame(3)))),
        (not(A == 4);(A == 4,mode_frame_(U))),
        write('Frame selected'),nl,
        fail,!.

choice_frame :-
  write('*******************************'),nl,
  write('*                             *'),nl,
  write('*        EXPERIMENTAL  FRAMES           *'),nl,
  write('*                             *'),nl,
  write('*   Please select the experimental frame that is be   *'),nl,
  write('*   used :                        *'),nl,
  write('*     1. - multiple runs with single observations     *'),nl,
  write('*     2. - single long run with multiple observations  *'),nl,
  write('*          and transient periods between observations  *'),nl,
  write('*     3. - single long run with multiple sequential    *'),nl,
  write('*          observations                     *'),nl,
  write('*     4. - do not know                      *'),nl,
  write('*                             *'),nl,
  write('*******************************'),nl,!.
```

Appendix 11 - b : Listing of EXPERT rule File

```
/* ********************************************************************** */
/* ********************************************************************** */
/* ********
/* ********  Expert Rules for Optimisation : Alternative 1
/* ********
/* ********************************************************************** */
/* ********************************************************************** */

/*  These rules are to replace the default rules proposed in WES.  The
/*  expertise portrayed in this file uses empirical evidence found from
/*  experimentation.
/*  Translation of these rules may be found in chapter 7.


quest(1101) :- write(' ? ').
quest(1102) :- write(' ? ').
quest(1103) :- write('What queue designates this service ? ').
quest(1104) :- write('  (Percentage please) :  ').
quest(1105) :- write('Establish all the connections now ? ').

own(optim_(1)) :-
        comtry(1),
        retract(comtry(1)),
        pred_rule_(Y),
        runten,!.

own(optim_(2)) :-
        (ecritu(6);(not(ecritu(6)),assert(ecritu(6)))),
        write('Using expert 1 rules '),nl,
        retractall(pemove(_,_,_,_,_)),
        retractall(pdecision(_,_,_,_)),
        fail,!.

own(optim_(3)) :-
        expnumber(Hh),
        X is Hh-1,
        wfac(100,X,U,Z),
        wfac(200,R,S,['Best performance ']),
        U > S,
        retract(wfac(200,R,S,['Best performance '])),
        assert(wfac(200,X,U,['Best performance '])),
        fail,!.

own(optim_(4)) :-
        pnamenu(Y),
        Y == 1,
        assert(action(0)),!.
```

```
own(optim_(5)) :-
        pnamenu(Y),
        Y > 1,
        not(wfac(100,R,S,T)),
        assert(wfac(200,0,0,['Best performance '])),
        find_connections,
        optexp,
        write('Using expert 1 rule 5 '),nl,
        eval_rule_(X),!.

own(optim_(6)) :-
        pnamenu(Y),
        Y > 1,
        expnumber(En),
        E is En-1,
        count(A),
        A < Y,
        own(qustatus(A,E)),
        write('Using expert 1 rule 6 '),nl,
        fail,!.

own(optim_(7)) :-
        pemove(Number,En,Va,Vb,10),
        write('Using expert 1 rule 7 '),nl,
        eval_rule_(X),!.

own(optim_(8)) :-
        pnamenu(Y),
        count(A),
        A < Y,
        own(pdecision(A)),
        Aa is Y-1,
        A == Aa,
        pdecision(_,_,_,_),
        write('Using expert 1 rule 8 '),nl,
        eval_rule_(X),!.

own(optim_(9)) :-
        pnamenu(Y),
        Y > 1,
        expnumber(En),
        E is En-1,
        count(A),
        A < Y,
        own(enstatus(A,E)),
        write('Using expert 1 rule 9 '),nl,
        fail,!.

own(optim_(10)) :-
        pemove(Number,En,Va,Vb,10),
        write('Using expert 1 rule 10'),nl,
        eval_rule_(X),!.

own(optim_(11)) :-
        pnamenu(Y),
```

```
        count(A),
        A < Y,
        own(pdecision(A)),
        Aa is Y-1,
        A == Aa,
        pdecision(_,_,_,_),
        write('Using expert 1 rule 11 '),nl,
        eval_rule_(X),!.


own(optim_(12)) :-
        write('Using expert 1 rule 12 '),nl,
        assert(action(0)),!.


own(pdecision(A)) :-
        retractall(prob(_)),
        expnumber(E),
        own(add(A,E)),
        prob(Prob),
        own(decision(A,E,Prob)),!.

own(pdecision(A)).

own(decision(A,En,Prob)) :-
        pname(A,[Name],De,Lo,Up),
   nl,write('Considering '), write(Name), write(' with evidence : '),
        write(Prob),nl,
        Prob > 6,
        pmove(A,En,Va,Vb,Ga),
        own(check(A,En)),
        Va < Vb,
        retractall(pstatus(A,En,_)),
        asserta(pstatus(A,En,Vb)),
        write('Expert has decided to set parameter '),
        write(Name),
        write(' = '),
        write(Vb),
        write('      Evidence : '),
        write(Prob),nl,
        asserta(pdecision(A,En,Vb,Prob)),!.

own(decision(A,En,Prob)) :-
        Prob < -6,
        pmove(A,En,Va,Vb,Ga),
        own(check(A,En)),
        Vb < Va,
        retractall(pstatus(A,En,_)),
        asserta(pstatus(A,En,Vb)),
        pname(A,[Name],De,Lo,Up),
        write('Expert has decided to set parameter '),
        write(Name),
        write(' = '),
        write(Vb).
```

```
        write('      Evidence : '),
        write(Prob),nl,
        asserta(pdecision(A,En,Vb,Prob)),!.

own(decision(A,En,Prob)).

own(check(A,En)) :-
        pmove(A,En,Va,Vb,Ga),
        not(pmove(A,E,Vb,Va,Gb)),!.

own(check(A,En)) :-
        pmove(A,En,Va,Vb,Ga),
        pmove(A,E,Vb,Va,Gb),
        ((Ga < 0 , Gc is -Ga, Gc >= Gb) ; (Ga >= 0, Gc is -Gb, Gc > Gb)),!.


own(add(A,E)) :-
        asserta(prob(0)),
        pmove(A,E,Va,Vb,Prob),
        own(tot(Prob)),
        fail,!.

own(add(A,E)).

own(tot(Prob)) :-
        prob(S),
        R is S+Prob,
        retract(prob(S)),
        asserta(prob(R)),!.


own(qustatus(A,E)) :-
        write('Check 1 ... '),
        qustatus(A,[Qname],E,Me,Lo,Up,_),
        write(Qname),nl,
        Me > 95,
        aconnection(1,[Qname],[Pname],10),
        pname(Nu,[Pname],De,Le,Ue),
        pstatus(Nu,E,Va),
        Vb is Va+2,
        Vb =< Ue,
        En is E+1,
        retractall(pstatus(Nu,En,_)),
        asserta(pstatus(Nu,En,Vb)),
        write('Expert has decided to set parameter '),
        write(Pname),
        write(' = '),
        write(Vb),
        write('      Evidence : '),
        write(10),nl,
        assert(pemove(Nu,En,Va,Vb,10)),!.


own(qustatus(A,E)) :-
```

```
        write('Check 2 ... '),
        qustatus(A,[Qname],E,Me,Lo,Up,_),
        write(Qname),nl,
        Me > 90,
        aconnection(1,[Qname],[Pname],10),
        pname(Nu,[Pname],De,Le,Ue),
        pstatus(Nu,E,Va),
        Vb is Va+1,
        Vb =< Ue,
        En is E+1,
        assert(pmove(Nu,En,Va,Vb,10)),!.

own(qustatus(A,E)) :-
        write('Check 3 ... '),
        qustatus(A,[Qname],E,Me,Lo,Up,_),
        write(Qname),nl,
        Me > 70,
        aconnection(1,[Qname],[Pname],10),
        pname(Nu,[Pname],De,Le,Ue),
        pstatus(Nu,E,Va),
        Vb is Va+1,
        Vb =< Ue,
        En is E+1,
        assert(pmove(Nu,En,Va,Vb,7)),!.

own(qustatus(A,E)) :-
        write('Check 4 ... '),
        qustatus(A,[Qname],E,Me,Lo,Up,_),
        write(Qname),nl,
        Me < 10,
        aconnection(1,[Qname],[Pname],10),
        pname(Nu,[Pname],De,Le,Ue),
        pstatus(Nu,E,Va),
        Vb is Va-2,
        Vb >= Le,
        En is E+1,
        assert(pmove(Nu,En,Va,Vb,-10)),!.


own(qustatus(A,E)) :-
        write('Check 5 ... '),
        qustatus(A,[Qname],E,Me,Lo,Up,_),
        write(Qname),nl,
        Me < 20,
        aconnection(1,[Qname],[Pname],10),
        pname(Nu,[Pname],De,Le,Ue),
        pstatus(Nu,E,Va),
        Vb is Va-1,
        Vb >= Le,
        En is E+1,
        assert(pmove(Nu,En,Va,Vb,-10)),!.


own(qustatus(A,E)) :-
        write('Check 6 ... '),
```

```
        qustatus(A,[Qname],E,Me,Lo,Up,_),
        write(Qname),nl,
        Me < 35,
        aconnection(1,[Qname],[Pname],10),
        pname(Nu,[Pname],De,Le,Ue),
        pstatus(Nu,E,Va),
        Vb is Va-1,
        Vb >= Le,
        En is E+1,
        assert(pmove(Nu,En,Va,Vb,-7)),!.


own(enstatus(A,E)) :-
        write('Check 7 ... '),
        noresult(1,[Pna]),
        enstatus(A,E,Me,Lo,Up,_),
        ename(A,[Ename],Ax,Bz,Cz),
        write(Ename),nl,
        Me > Up, not(Up == 0),
        aconnection(2,[Ename],[Pname],P),
        not(Pname == Pna),
        aconnection(1,[Qname],[Pname],10),
        qustatus(No,[Qname],E,Mi,Li,Ui,_),
        Mi > 70,
        pname(Nu,[Pname],De,Le,Ue),
        pstatus(Nu,E,Va),
        Vb is Va+1,
        Vb =< Ue,
        En is E+1,
        assert(pmove(Nu,En,Va,Vb,10)),!.


own(enstatus(A,E)) :-
        write('Check 8 ... '),
        noresult(1,[Pna]),
        enstatus(A,E,Me,Lo,Up,_),
        ename(A,[Ename],Ax,Bz,Cz),
        write(Ename),nl,
        Me > Up,
        aconnection(2,[Ename],[Pname],P),
        not(Pname == Pna),
        aconnection(1,[Qname],[Pname],10),
        qustatus(No,[Qname],E,Mi,Li,Ui,_),
        Mi > 60,
        pname(Nu,[Pname],De,Le,Ue),
        pstatus(Nu,E,Va),
        Vb is Va+1,
        Vb =< Ue,
        En is E+1,
        assert(pmove(Nu,En,Va,Vb,P)),!.



own(enstatus(A,E)) :-
        write('Check 9 ... '),
```

```
        noresult(1,[Pna]),
        enstatus(A,E,Me,Lo,Up,_),
        ename(A,[Ename],Az,Bz,Cz),
        write(Ename),nl,
        Me < Lo,
        aconnection(2,[Ename],[Pname],P),
        not(Pname == Pna),
        aconnection(1,[Qname],[Pname],10),
        qustatus(No,[Qname],E,Mi,Li,Ui,_),
        Mi < 50,
        pname(Nu,[Pname],De,Lo,Up),
        pstatus(Nu,E,Va),
        Vb is Va-1,
        Vb >= Le,
        En is E+1,
        Pp is -P,
        assert(pmove(Nu,En,Va,Vb,Pp)),!.

aconnection(1,[Qname],[Pname],P) :- connection(1,[Qname],[Pname],P),!.

aconnection(1,[Qname],[Pname],Ga) :-
  write('Which parameter designates the number of resources'),nl,
  write(' for the service '),
  write(Qname),
  ask_con,
  answer(1101,[Pname]),
  Ga is 10,
  asserta(connection(1,[Qname],[Pname],Ga)),!.


aconnection(2,[Ename],[Pname],P) :- connection(2,[Ename],[Pname],P),!.

aconnection(2,[Ename],[Pname],P) :-
        aconnection(3,[Ename],[Qname],P),
         noresult(1,[Pnama]),
       not(Qname == Pnama),
         aconnection(1,[Qname],[Pname],Pa),!.

aconnection(2,[Ename],[Pname],P) :-
        aconnection(3,[Ename],[Qname],P),
         noresult(1,[Pname]),
      asserta(connection(2,[Ename],[Pname],10)),!.


aconnection(3,[Ename],[Qname],P) :- connection(3,[Ename],[Qname],P),!.

aconnection(3,[Ename],[Qname],P) :-
  nl, present_queues, nl,
  write('Is there a service that entities of type '),
  write(Ename),
  write(' use most'),
  question(1102),
  answer(1102,[E]),
  (E == y ; E == yes),
  ask_con1,
```

```
 answer(1103,[F]),
 own(use([F],[Ename],Ga)),
 asserta(connection(3,[Ename],[F],Ga)),
 P is Ga,!.

aconnection(3,[Ename],[Qname],P) :-
 answer(1102,[E]),
 (E == n ; E == no),
 noresult(1,[Qname]),
 asserta(connection(3,[Ename],[Qname],10)),!.

ask_con1 :-
 question(1103),
 answer(1103,[Qname]),
 qname(Nz,[Qname],Dz,Lz,Uz),!.

ask_con1 :-
 write(' Sorry, I am not aware of this service queue name'),nl,
 ask_con1,!.

ask_con :-
 question(1101),
 answer(1101,[Pname]),
 pname(Nz,[Pname],Dz,Lz,Uz),!.

ask_con :-
 write(' Sorry, I am not aware of this parameter name'),nl,
 ask_con,!.

own(use([F],[Ename],Ga)) :-
        write('Please indicate the relative usage of service '), write(F),nl,
        write(' by entities of type '), write(Ename),
        question(1104),
        answer(1104,[G]),
        Ga is G // 10,!.

own(use([F],[Ename])).

find_connections :- question(1105),
        answer(1105,[E]),
        (E == y ; E == yes),
        conqp,
        conep,!.

find_connections.

conqp :- qnamenu(Y),
        count(A),
        qname(A,[Qname],Za,Zb,Zc),
        aconnection(1,[Qname],[Pname],Pa),
        Aa is Y-1,
        A == Aa,!.

conqp :- write(' Cannot find all the service connections ... '),nl,!.
```

```
conep :- qnamenu(Y),
        count(A),
        ename(A,[Ename],Za,Zb,Zc),
        aconnection(2,[Ename],[Pname],Pa),
        Aa is Y-1,
        A == Aa,!.

conep :- write(' Cannot find all the entity connections ... '),nl,!.


noresult(1,['no_connection']).
```

Appendix 12

WES and USER Experiment

This appendix presents some of the material involved in the experimentation involving 20 volunteers. The following are presented:

- part 1 of the experiment;
- part 2 of the experiment;
- results.

SIMULATION EXPERIMENT - May 1988

Dear

Thank you for taking part in this experiment. It involves the use of a simulation model and will require a couple hours of your time and some common sense. The objective is to try and assess your ability at using simulation intelligently despite minimum knowledge about this technique. Your results will used in aggregate form as experimental evidence in my doctoral thesis.

Please proceed sequentially through these sheets. Instructions are provided as you go along.

Please complete :

Your name :

Your course and Year (or job title) :

The date :

The time now :

Finish time :

Have you ever used simulation before ?
If yes, please give details :

Experiment Overview

1. Demonstration
2. Knowledge Questionnaire
3. Problem Description
4. Initial Approach
5. Experimentation
6. Results
7. WES
8. Refinement
9. New Results
10. Feedback

1. Demonstration : First you will be given an introductory tutorial to visual simulation to provide you with the basic ideas involved.
2. Knowledge Questionnaire : Second, you will be asked to answer a series of questions relating to simulation.
3. Problem Description : Third, you will be presented with the problem that you are to address.
4. Initial Approach : Fourth, you will be asked to explain how you propose to go about this problem.
5. Experimentation : Fifth, you will be given the opportunity to carry out your investigation.
6. Results : Sixth, you will be asked to summarise your conclusions and the reasons for your decision.
7. WES : Seventh you will introduced to an experimentation facility.
8. Refinement : You will be invited to improve your experiments.
9. New results : You will be asked to summarise your conclusions in the light of possible new evidence.
10. Feedback : Finally tenth, in a couple of days I will invite you back for a brief postmortem.

1. Demonstration

You will now be given an introduction to simulation. Please type DEMOSIM on the keyboard and follow instructions. After the demonstration tutorial please proceed to step 2.

2. Knowledge Questionnaire

Please answer the questions on this sheet. Your answers do not have to be very long. Do not worry, if you cannot give the correct answer.

Why use simulation ?

How would you use simulation ?

What points would you need to consider carefully ?

Where is the place of statistics in simulation ?

Why replicate experiments ?

List any dangers you may think of about simulation :

What is :

- a steady state ?
- a run in time ?
- a model ?
- a run length ?
- significance of results ?
- utilisation of service ?
- time in the system of entities ?

3. Problem Description

Please read the following problem described in the next few pages.

You may look at the model by typing ARGON

When you have seen enough of the model, please proceed to the next step.

Problem Description

Problem Background

A company ARGON have a UK chain of retail stores. As a recent MBA graduate you have been asked to investigate staffing levels at one of their major stores particularly on busy Saturday mornings. There appears to be excessive delays for customers and a recommendation staffing levels is required urgently.

The Argon Store

Six types of customer use the store. They are known as 'cheq', 'cash', 'cred', 'view', 'jewl', and 'retn'. The 'cheq', 'cred', and 'cash' customers arrive randomly at the shop and after ordering their requirements proceed to separate payment counters where they pay by cheque, cash and credit card. These payment times are different for each payment type but all follow poisson distributions.

The 'cheq', 'cash', and 'cred' customers then proceed to the collection area where they collect their purchases before leaving the shop. Customers known as 'view', also arrive randomly, join the order queue, but only view the catalogues before leaving.

The store also operates a specialist jewellery service. Customers, known as 'jewl', (arriving randomly), go direct to the collections counter where they receive priority service for their jewellery purchases. The final type of customer 'retn' are those customers who wish to return previously purchased items. After entering the store, they move directly to the collections queue for their return and refund activity.

All customer arrivals, payment and service activities can be described by poisson processes. The table overleaf gives the average time for each activity together with the current level of staff at each service facility within the store. These times are believed to be accurate for the Saturday morning 'busy' period which typically occurs between 9.00 am and 1.00 pm. There is no mobility of staff within the store, i.e. staff working at the jewellery counter cannot move to (say) the credit payment counter.

You may make any reasonable assumptions not described above (including costs). The objective of the assessed work is to suggest more realistic operating staff levels.

CUSTOMER ARRIVAL TABLE	
Customer Type	Average inter arrival time (seconds)
Cash	150
Cheque	150
Credit	225
Non-order (view)	300
Returns	600
Jewellery	375

POISSON SERVICE TABLE	
Service Type	Average Service Time (seconds)
Ordering (and viewing goods)	180
Cash payments	180
Cheque payments	270
Credit payments	300
Collection (and returns)	75
Jewellery customers	375

STAFF RESOURCES	
Location	Number
Order Taking	2
Cash Collection	2
Taking Cheques	1
Handling Credit-cards	3
Jewellery/Goods out	4

Type ARGON to load the program.

A schematic layout of the ARGON store will appear on the screen with an INTERACTION box appearing in the top right hand corner of the screen.

Type RUN to run the model.

You will see the model operating (slowly).

The symbols 'cheq', 'cash', 'cred', 'view' will scroll down the screen on the left hand side while 'jewl', and 'rstr' appear on the right hand side of the screen. The symbols represent the different types of customer passing through the store.

Press the space bar to interact with the model. The model will stop and the INTERACTION box will again appear on the screen.

The responses to INTERACTION are:

RUN (to continue running the model.)

SPEED (to change the display speed of the model: initially it is set at 100. Once you understand the logic of the model you may wish to increase the speed to 200.)

BATCH (the graphics/animation helps to confirm the logic of the model. The BATCH command will turn off the animation so the model will run quickly and can be used in an experimental mode.)

STOP (to stop the model and return to the operating system.)

OWN (to change the configuration parameters of the store.)

Typing OWN will bring a menu on to the screen so that the configuration of the shop can be changed,

The model uses a time unit of 1 second and the menu shows the average interarrival times of all the different customers. As an example

1: A. R. CASH CUSTOMERS 150

indicates that cash customers arrive randomly but with an average time between customer arrivals of 150 seconds. This average interarrival time can be changed by typing 1 followed by the new average. This same method is used for all menu options.

The other options on this first menu are:

8: RANDOMISE

(A simulation model is an experiment RANDOMISE will change the random number sequences. Type a value such as 25 to get different starting positions for all the model's random number streams.)

9: CHANGE SERVING RATES

Will enable any average serving rate to be changed.

10: CHANGE NO OF SERVERS

Will enable the number of staff serving at each section of the shop to be changed.

11: VIEW HISTOGRAMS

This interaction will enable you to view six histograms which give the time in the shop for all customer types.

12: CLEAR ALL HISTOGRAMS

Will empty all histograms so that a new experiment can be started.

4. Initial Approach

Please outline how you would go about addressing this ARGON store problem. Describe any experiment you may think appropriate. Mention points you believe are important. Consider the criteria you would use.

5. Experimentation

By following the instructions provided in the problem description sheet, please carry out your own experimentation. Use the sheet below to record the experiments that you are carrying out. Note any comments.
When you are satisfied with your results please proceed to the next step.

E-periment number			
Parameters			
order servers			
cheque servers			
cash servers			
credit servers			
collect servers			
Utilisation of service			
order_q			
cheque_qu			
cash_qu			
credit_qu			
collect_q			
level_q			
Time-in-the-system entities			
cash customers			
cheque customers			
credit customers			
[illegible] customers			
[illegible] customers			
[illegible] customers			

6. Results

Please indicate your recommendations : Configuration :

Why ?

Extra considerations?

Any Remarks about this experiment ?

7. WES (Warwick Expert Simulator)

I would now like to introduce you to a tool that allows you to perform formal experimentation.
Please type DEMOWES.
After the demonstration, please proceed to the next step.

8. Experimentation

By using the WES system, please refine your own investigation into the problem. You may access WES by typing RICK.

Use the sheet below to record the experiments that you are investigating. When you are satisfied with your results, please proceed to the next step.

Experiment number			
Parameters			
order servers			
cheque servers			
cash servers			
credit servers			
collect servers			
Utilisation of service			
order_q			
cheque_qu			
cash_qu			
credit_qu			
collect_q			
Time in the system entities			
cash_customers			
cheque customers			
credit customers			
[illegible] customers			
[illegible] customers			
[illegible] customers			

9. New Results

Please indicate you final recommendations : Configuration :

Why ? Are your results different than before ?

Extra considerations ?

Any Remarks about this experiment ?

Appendix 12 - b : Part 2 of the main experiment

SIMULATION EXPERIMENT (Part 2) - May 1988

10. Feedback

Dear

Thank you again for taking part in this experiment. In the final part of this experiment, you will not required to use a computer again, only to complete sequentially the following sheets.

Please complete :

Your name :

The date :

11. Knowledge Questionnaire

Please answer the questions on this sheet. Your answers do not have to be very long. Do not worry, if you cannot give the correct answer.

Why use simulation ?

How would you use simulation ?

What points would you need to consider carefully ?

Where is the piece of statistics in simulation ?

Why replicate experiments ?

List any dangers you may think of about simulation :

What is :

- a steady state ?
- a run in time ?
- a model ?
- a run length ?
- significance of results ?
- utilisation of service ?
- time in the system of entities ?

12. Consultancy Problem

Overview

Your cousin who works in Malaysia for their Immigration Office has just written to you for advice about the following problem.

He has just been given a week off his normal duties to sort out a staffing problem in one of the departments. Prior to his involvement, the Immigration Office had used the services of external consultants (BobbyVision & Co). These consultants had conducted an early simulation study into the problem. By the time their contract expired, BobbyVision & Co had developed a model of the problematic department. This model had been fully verified and validated, but little experimentation had been undertaken. The Immigration Office decided not to extend the contract judging that the model was sufficient for their own staff to resolve the existing staffing problems. And your cousin has been landed with the job.

His conclusions will be used in the department's budgeting for the coming year. He feels very anxious about this project and believes his job could be at stake if his conclusions are wrong. To add to his uneasy feeling about this task, he writes that he has never used simulation before!!

Your cousin is writing to you because he has heard that recently, you were involved with a simulation study using a model also developed by BobbyVision & Co. He enclosed a description of the problem; a list of the assumptions that operate the model as explained by one of the consultants in a memo; and a screen dump of the model after a run.

Please read the problem description and proceed to the next step.

1. Description of Problem

A typical Passport Application and Collection Unit in a Malaysian Immigration Department is investigated, and a simulation model is developed. A brief structure of the Unit is that, there is an Inquiry counter where the applicants obtain necessary information and application forms. There are three separate counters for different types of passports to be applied, that is, the counter for International passport, Local passport(passport needed to go to Singapore and East Malaysia only) and a *Special* counter to deal with applicants applying passports needed to go to China and other special cases. There is a waiting area for the applicants to wait for their passports to be collected. One more counter is available here for the applicants to collect passports.

There are four types of applicants entering to this unit. There are applicants applying for Local passports, applicants applying for International passports, *Special* cases applicants and applicants coming back to collect their passports(these are previously *Special* cases and International passports applicants who had received notices to come back to collect their passports). Every applicant arrives to this Unit will pass through the Inquiry counter. The applicants for passport application obtain application forms here and then get directed to the appropriate serving counters. The applicants come back for collecting passports hand in their acknowledgement card(a notice sent by the Department indicating passports are ready for collection) to the officer at the Inquiry counter, and then get directed to the waiting area.

The operational situation is that, each applicant enter the Unit must queue at the Inquiry counter to be served. When an inquiry officer is free, the applicant at the head of the queue will be served. Then, the applicant will move to the appropriate counter according to the type of passport he is applying for. Again, once an officer in the counter is free, the person entering the queue first will be served.

After finishing the passport application process, the applicants of Local Passport will go to the waiting area to wait for passports to be collected. The applicants of International Passports and Special cases will leave the Department until further notices to be sent. Once a applicant leaves the Passport Application and Collection Unit, his further motive will not be considered in this model.

Assumptions made in the model

(1) I have considered only the Passport Application Unit in the Department, which only deal with three types of applicants. Not the whole Immigration Department.

(2) It is assumed that no applicants will leave the Unit half way through the process(queue or service).

(3) It is assumed that no applicants apply more than one type of passport. Although this could be incorporated in the model by increasing the number of arrivals, ie. if some Local Passport applicants will also apply International Passport, the mean inter-arrival time for International Passport applicants would need to be increased.

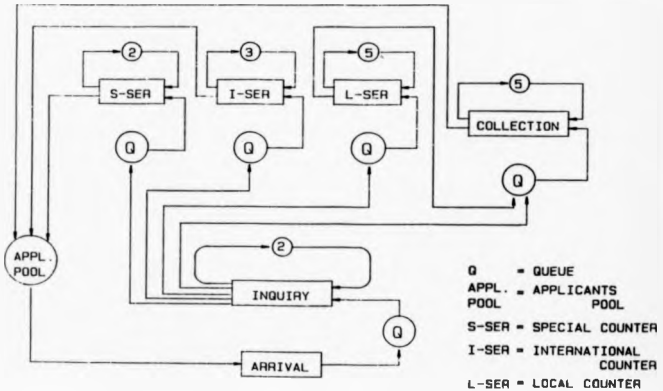
(4) The waiting area is considered as a queue in this model, although the applicants in this area do not have to queue up. It is assumed that the first applicant entering this waiting area will get his passport first. The assumption is made based on the facts that when the Unit first accept acknowledgement of the applicants entering the waiting area (either just completed the application process for Local Passports applicants, or just handed in the acknowledgement cards for applicants coming back for passports collection), will be called to collect their passport first. The number of servers in the waiting area means the internal officers who are dealing with processing passports inside the Unit, not serving at the counter. Once the passports are processed, they will be brought to the counter and the names of the passports holders will be called.

(5) It is assumed that if an Office in a particular counter is free, he/she will not help in the other counters.

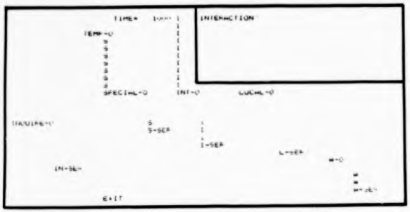
(6) It is assumed that the officers are on shift-duty, so that no tills will be closed suddenly.

(7) It is noted that the Department opens at 9:30 am

(8) The NEGATIVE EXPONENTIAL distribution is used for the arrival speed of Local, International and Passport Collection applicants. The POISSON distribution is used of the arrival speed of Special Cases applicants. The NORMAL distribution is used for all servicing time in all counters.



ENTITY AND ACTIVITY CYCLES




```
MENU OF SIMULATION OPTIONS :
 0: CONTINUE THE SIMULATION
 1: ARRIVAL SPEED (LOCAL)                          3
 2: ARRIVAL SPEED (INTERNATIONAL)                  5
 3: ARRIVAL SPEED (SPECIAL)                       20
 4: SERVING TIME (INQUIRY)                         2
 5: SERVING TIME (LOCAL)                          15
 6: SERVING TIME (INTERNATIONAL)                  20
 7: SERVING TIME (SPECIAL)                        30
 8: ARRIVAL SPEED (COLLECTION)                    15
 9: WAITING TIME (COLLECTION)                     20
10: NO. OF SERVERS (INQUIRY)                       2
11: NO. OF SERVERS (LOCAL)                         4
12: NO. OF SERVERS (INTERNATIONAL)                 3
13: NO. OF SERVERS (SPECIAL)                       1
14: NO. OF SERVERS (COLLECTION)                    8
15: RANDOMISE
16: SIMULATION SPEED (0--200)                    100
17: DISPLAY HISTOGRAMS                             0
  : 0 = MODEL : 1 = HISTOGRAMS
WHAT OPTION PLEASE ?
```

13. Report for Simulation experimentation

All your cousin has managed so far is to establish that the problem involves staffing 5 different counters for 4 different types of applicants.

The inquiry desk currently has 2 staff
The local counter has 4 staff
The international counter has 3 staff
The special counter has 1 staff
The collection desk has 6 staff

The different type of applicants are :

Local passport applicants
International applicants
Special case applicants
Collection applicants

Your recommendations to address the problem.

Please indicate to your cousin how he might go about addressing his problem. Outline a general approach.

Simulation experimentation

Since your cousin is so anxious about getting correct results, please indicate to him how he might spend some of his time in performing proper experiments. Suggest a possible experimental approach, and indicate areas where he must be careful. Make any other comment you may think appropriate to help him.

Appendix 12 - c : Results from the WES and USERs experiment

In this appendix, the results of the experiment described in section 7.6 are described.

Reference	Occupation	Background
csto	P	Engineer: electrical
drot	P	Engineer: astro-physics
skar	S	Engineering: electronics
dmou	S	Engineering: electronics
hpal	S	Law
erob	P	English : writer
kabs	P	History
mhas	P	management science
msol	S	mathematics
fahm	P	accountant
stay	S	biological sciences
iard	P	accountant
sdav	P	mathematics
nver	S	Engineering: civil
agri	S	Engineering: civil
fjon	S	Sociology
cmit	S	Engineering: civil
aabr	P	Philosophy - Store manager
thod	P	Computer consultant
rtom	P	Computer consultant & research associate
mila	P	Computer consultant & research associate

Table a12.1 : Subjects and their background.

Ref	K Score	Initial Qt	Fa	Ob	ARGON EXPERIMENTS N	Configuration	Aw	WES Experiments N	Configuration	Changed	NP
cato	9	n	n	y	4	3 1 3 3 3	v				
drot	15	y	n	y				4	4 2 2 2 3		
skar	9	n	n	y				4	4 3 3 2 3		
dmou	14	y	n	y				4	4 2 2 3 4		
hpal	10	n	n	y	4	5 3 2 2 3	y	5	4 2 2 2 3	y	2
erob	10	y	n	y	3	5 2 2 2 4	n	4	5 2 2 2 3	y	1
kabs	8	y	n	y	4	5 4 3 2 4	n	3	4 2 2 2 3	y	4
mhas	12	y	y	y	3	5 3 3 3 4	y	3	4 2 3 2 3	y	4
msul	12	n	n	y	5	4 2 3 2 3	n	2	4 2 2 2 3	y	1
fahm	10	n	n	y	3	4 4 4 4 4	n	3	4 2 3 2 4	y	3
stay	11	y	n	y	3	6 2 2 3 6	n	2	4 2 3 2 3	y	4
iard	11	n	n	y	9	4 2 3 2 5	n	2	4 2 2 2 3	y	2
sdav	6	y	n	y	3	5 3 3 3 6	n	3	4 2 3 2 3	y	4
cmit	10	y	n	y	4	7 4 5 2 4	n	3	4 2 2 2 3	y	4
fjon	8	n	n	y	4	6 3 2 2 5	n	3	5 2 3 2 3	y	4
nver	15	y	n	y	4	6 2 2 2 4	n	2	5 2 2 2 3	y	2
agri	4	y	n	y	4	6 2 4 2 5	n	3	5 2 3 2 3	y	3
aabr	4	n	n	y	3	4 3 2 3 4	n	3	5 2 3 3 3	y	4
thod	14	y	y	y	5	4 2 3 3 3	y	3	5 2 3 2 4	y	3
rlom	12	y	n	y	2	4 3 1 3 4	n	3	5 2 3 4 4	y	4
msla	8	y	n	y	4	5 3 2 1 3	n	3	5 2 3 3 3	y	3

where :

Ref : subject reference
K Score : score of knowledge questionnaire
Initial:
Qt: make use of queuing theory
Fa: prepare formal experimentation
Ob: use observations
N : number of experiments
Configuration : recommendation for parameter values
Aw : diplay any awareness about limitations of results
WES N : number of experiments using WES
WES configurations : recommendations using WES for paramater values
Changed : did the recommendations change.
NP : number of parameters changed.

Table a12-2 : Results from users

Analysis

	Before	After
Order		
	3 : *	3 :
	4 : * * * * * * *	4 : * * * * * * * * * * * *
	5 : * * * * * * *	5 : * * * * * * * *
	6 : * * * *	6 :
	7 : *	7 :
	4 or 5	4
Cheque		
	1 : *	1 :
	2 : * * * * * * * *	2 : * * * * * * * * * * * * * * * * * *
	3 : * * * * * * * *	3 : *
	4 : * * *	4 :
	2 or 3	2
Cash		
	1 : *	1 :
	2 : * * * * * * * *	2 : * * * * * * * * * *
	3 : * * * * * * * *	3 : * * * * * * * * * * *
	4 : * *	4 :
	5 : *	5 :
	2 or 3	3
Credit		
	1 : *	1 :
	2 : * * * * * * * * * *	2 : * * * * * * * * * * * * * * *
	3 : * * * * * * *	3 : * * *
	4 : *	4 : *
	2	2
Collect		
	3 : * * * * *	3 : * * * * * * * * * * * * * * * *
	4 : * * * * * * * *	4 : * * * *
	5 : * * *	5 :
	6 : * *	
	4	3

Table a12-3 : Results of parameter value selection

Knowledge and Awareness Issues

Results were collected relating to the knowledge and awareness that might have been acquired by non-knowledgeable users through the experience of using WES. Unfortunately, it was not possible to obtain results from all the subjects.

Ref	K Score	Initial Qt	nK Fa	Suggested Ob	Score	Qt	Fa	Ob
skar	9	n	n	y	15	y	y	y
mhas	12	y	y	y	11	y	y	y
stay	11	y	n	y	9	y	n	y
iard	11	n	n	y	13	y	y	y
msla	8	y	n	y	11	y	y	y
fatm	10	n	n	y	12	y	y	y
droth	15	y	n	y	18	y	y	y
agri	4	y	n	y	12	y	n	y
nver	15	y	n	y	16	y	y	y
thod	14	y	y	y	11	y	y	y

where :
- Ref : is the reference identification of the subject
- K Score : is knowledge score from initial attempt at the questionnaire
- Initial : covers the period prior to exposure to WES (for ARGON model)
  - Qt relates whether users initially considered queueing theory
  - Fa relates whether users initially considered formal analysis
  - Ob relates whether users initially considered observation of visual model
- nK Score : is knowledge score from questionnaire after the experiment
- Suggested : covers the period after using WES (IMMIGRATION OFFICE model)
  - Qt relates whether users initially considered queueing theory
  - Fa relates whether users initially considered formal analysis
  - Ob relates whether users initially considered observation of visual model

Table 12-4 : Results from knowledge and awareness increase by users after using WES

Increase knowledge score :
YES : 70%
NO : 30%

Table 12-5 : Percentage number of subjects that recorded increase in knowledge test score

Increase awareness :
YES : 60%
NO : 20%
No possible increase : 20%

Table 12-6 : Percentage number of subjects that showed increased awareness

Appendix 13

Consultations with WES

The objective of the following appendices is to present some of the features of WES through consultation experience. In each consultation, a different task of WES will be reviewed and prior to their presentation, it will be indicated which aspects can be seen.

Appendix 13-a : Consultation using a retail store simulation model

Features

This consultation presents:

- the prediction task
- the evaluation task
- use of extra files of rules
- automatic model understanding by WES of WITNESS models
- experimentation in WITNESS environment

Background:

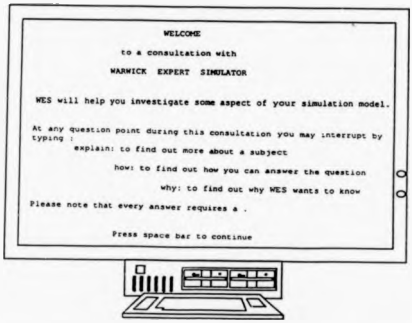
This model is a WITNESS version the ARGON problem presented in chapter 7.

Consultation

The WES system is loaded by typing 'WITWES'.

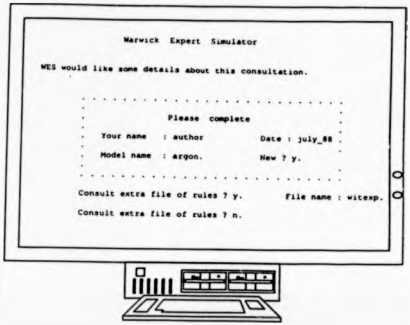
The focus of interest lies in a consultation using the WITNESS package. This is requested by moving the cursor (*) to the appropriate selection.

The user is introduced to the WES system consultation pattern.



WES stores results and consultation summaries. It is therefore useful to obtain some indication about who was performing the consultation and when. WEs also takes this opportunity to find

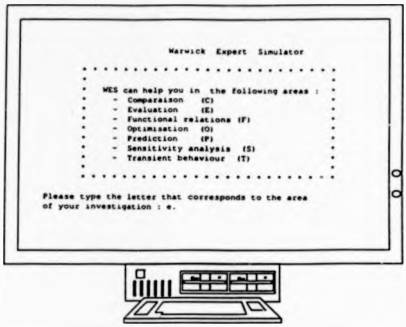
out which model is to be considered. This obtained from the user below.



Several things have been asserted. Apart from the name of the user and the date of this consultation, WES knows that it is about to deal with a model for which it posses no knowledge. It will have to learn about it. Also, WES has been told that it is to use rules that are held in file "witexp". This happens to be the files of rules for use with the WITNESS package.

WES has read the WITNESS datafile for the model called ARGON. It has shared its understanding investigation with the user. It will not be necessary to perform this task next time since a file will be saved with all this information.

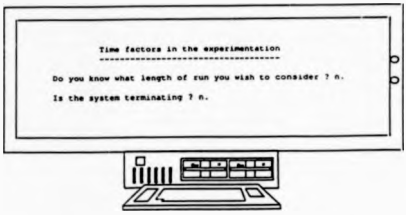
The consultation may now begin properly.



In this example, the evaluation task is requested. It was argued in chapter 7, that the use of this task with WITNESS models provides a useful way for a user to learn about a model he knows very little about.

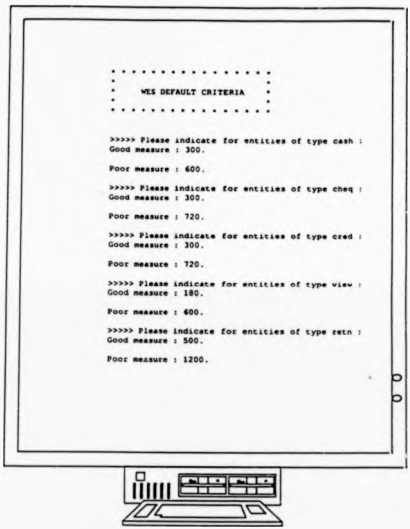
The user is offered a choice about the level of detail that he may request about this evaluation exercise. A quick summary evaluation simply provides a quantification value and qualification statement about the system's current performance. This in fact never proved very useful. A detailed evaluation provides additionally presentation of typical value for the quantification vari-

ables of the model.



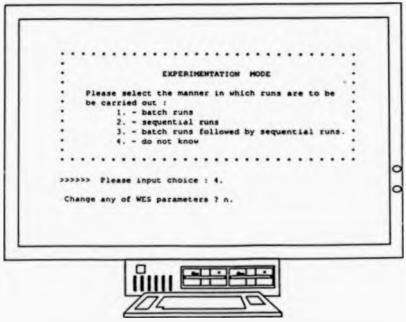
Time factors refer to the length of runs and the transient periods. Since the user is considered ignorant of this model, he fails to give WES any indication about these facts.

The default criteria is used which a sensible choice in a state of ignorance about a model.



Since the default criteria has been selected, the opportunity has been taken by WES to ask the user what expectation might be for good or bad performance. It is in fact possible for the user to answer 0 which the user will interpret as a fact not yet established. The rationale behind this approach is that it permits WES to draw the attention of the user if his expectation are not met. The user may either refine his understanding of the model, or tell WES to change the boundary values of acceptance.

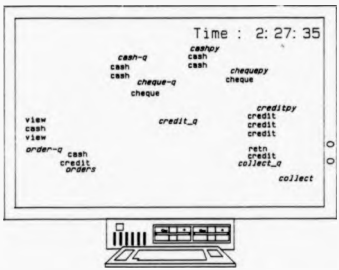
The user may select the experimental frame that he wishes to use. The default when using WTT-NESS is option 2.



In this example, the user requests that he does not know which experimentation mode to use. The default is generally set to option 3.

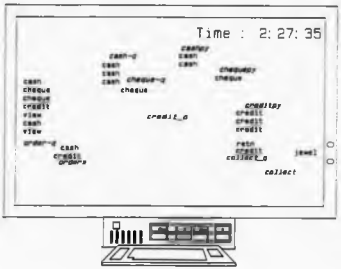
The question referring to the parameters of WES concerns the defaults of the system. They are presented elsewhere.

WES has activated a frame for determining an appropriate run in time. Control now goes to the WITNESS component where the model will be run in accordance with the specification of this frame.



Results from this monitoring exercise are read back and analysed. Once a suitable run in time has been found, WES may complete the design of the original experiment relating to evaluating the system.

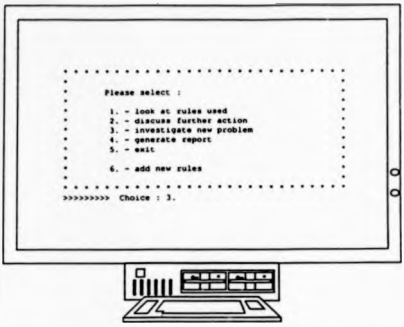
Operations return to WITNESS whilst this experiment is executed.



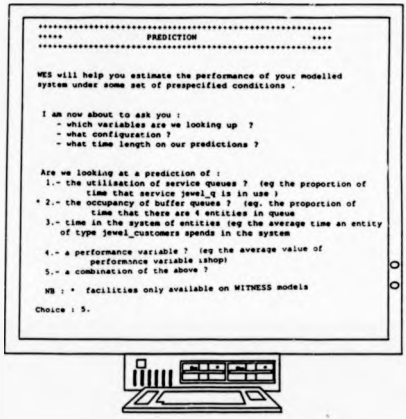
Results are read back.

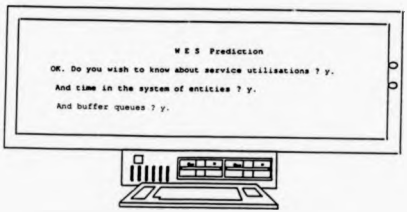
This process is repeated for each set of observations. The results are then analysed and presented to the user

The user is presented with an evaluation of his model. He has now gained an understanding of the different components in the model and an evaluation of their average quantification values.



In this example consultation, the user is assumed to wish to further focus his attention on the activities of cash paying customers over a twenty minute period.


PREDICTION
WES will help you estimate the performance of your modelled system under some set of prespecified conditions .
I am now about to ask you :
- which variables are we looking up ?
- what configuration ?
- what time length on our predictions ?
Are we looking at a prediction of :
1.- the utilisation of service queues ? (eg the proportion of time that service jewel_q is in use)
* 2.- the occupancy of buffer queues ? (eg. the proportion of time that there are 4 entities in queue
3.- time in the system of entities (eg the average time an entity of type jewel_customers spends in the system
4.- a performance variable ? (eg the average value of performance variable ishop)
5.- a combination of the above ?
NB : * facilities only available on WITNESS models
Choice : 5.


W E S Prediction
OK. Do you wish to know about service utilisations ? y.
And time in the system of entities ? y.
And buffer queues ? y.

The queue utilisation process of service cashpy is selected.

This way all the quantification variables relating to cash-customer process have been selected.

Do you wish to use all the default values of the parameters ? y.

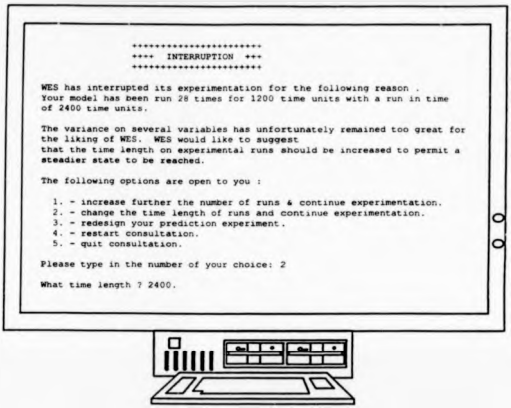
The same configuration is to be used.

This time the user has specified that the predictions are to be over a twenty minute period.

Notice that this time the user is not asked about the frame to use or the mode for collecting results. This is because WES has already obtained the user preference to these issues. However, the question relating to WES parameters allows a user to change the current settings.

WES was able to specify the experiment immediately: it did not need to fire the frame to detect a suitable run in time, since the configuration is the same has the same experiment and it had already established 2400 time units as an appropriate transient period between observations.

As before, control would go to WITNESS for the execution of the model. When results are read back and analysed WES finds that it cannot obtain satisfactory results.



WES will has already attempted some variance reduction techniques, since it states that the model has already been run 28 times, instead of the original 14 specified. In this example, the user decides to look at the cash-customer process over a 40 minute period. WES will return and design a new experiment.

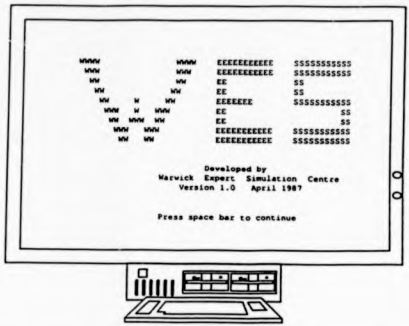
No experimentation needed to be run, since WES already had the desired results. Note the way, WES brings to the attention of the user extra facts that are not directly relevent to this experiment but affect the performance of the whole system.

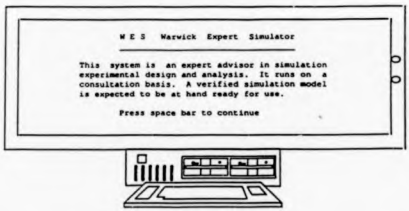
The user is assumed to have had enough and chooses to exit. Although he does not request a report, WES will prepare its own file with all the discovered facts of this experiment asserted.

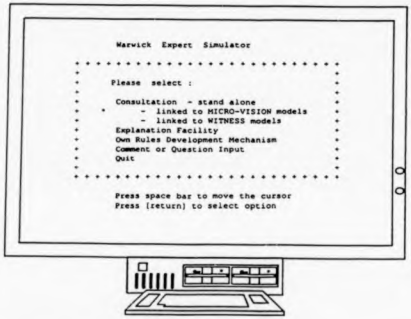
Appendix 13 - b : Consultation 2 using Laundrette Simulation model

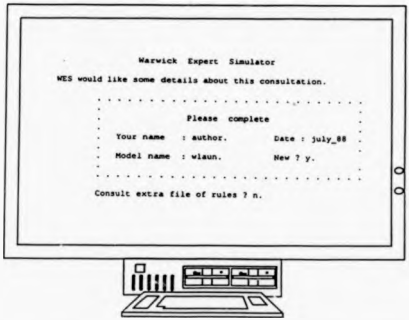
In this appendix, features of WES are presented about:

- the comparison task,
- the understanding of MICROVISION models via the user


WWWW WWWW EEEEEEEEEEE SSSSSSSSSS
WWW WWW EEEEEEEEEEE SSSSSSSSSS
WW WW EE SS
WW WW EE SS
WW W WW EEEEEEE SSSSSSSSS
WWW W WWW EE SS
WW WWW WW EE SS
WWW WWW EEEEEEEEEEE SSSSSSSSSS
WW WW EEEEEEEEEEE SSSSSSSSSS
Developed by
Warwick Expert Simulation Centre
Version 1.0 April 1987
Press space bar to continue


W E S Warwick Expert Simulator
This system is an expert advisor in simulation experimental design and analysis. It runs on a consultation basis. A verified simulation model is expected to be at hand ready for use.
Press space bar to continue

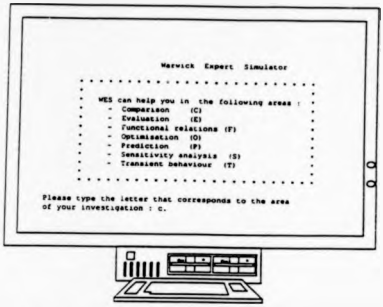

Warwick Expert Simulator
Please select :
Consultation - stand alone
- linked to MICRO-VISION models
- linked to WITNESS models
Explanation Facility
Own Rules Development Mechanism
Comment or Question Input
Quit
Press space bar to move the cursor
Press [return] to select option



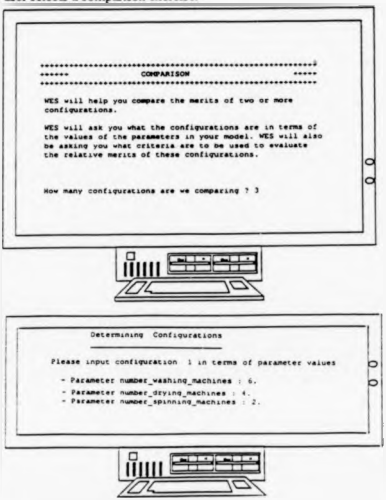
WES has been informed that it is a new model. It will need to obtain from the user details about the components in the model.

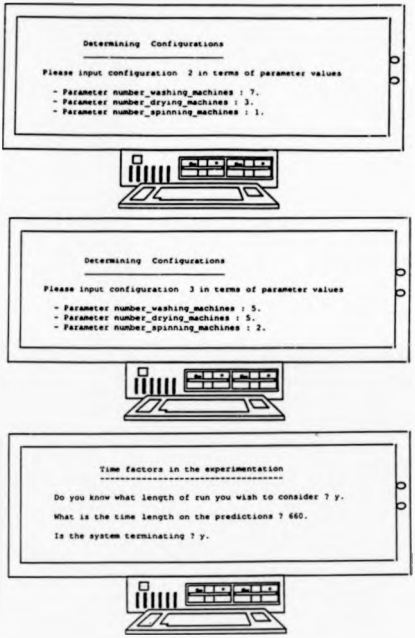
From this execise, WES will have obtained a feel for the components in the model. There is only

one seed and no user defined performance measures.



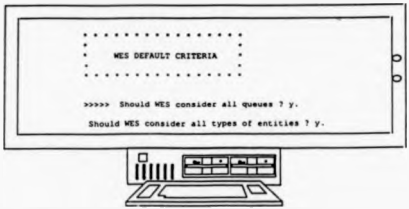
The user selects a comparison exercise.



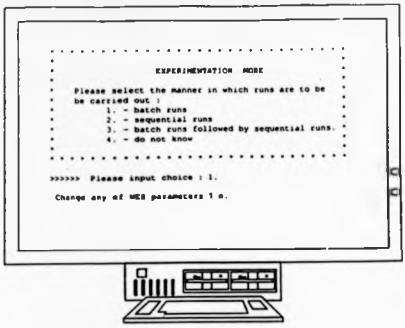


The user defines his laundrette as working an 11 hour day. Since the system is terminating, there will be no run in time.

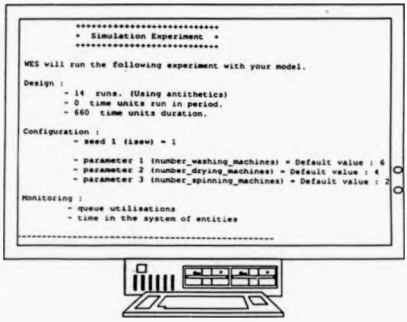
The user selects the criterion dealing with minimising the time-in-the-system of entities. This time he is concerned about the amount of time, customers spend in his shop.



When it is not an evaluation exercise of the whole model, the user is asked if all components should be considered. This allows a user to concentrate on a subprocess if he wishes to.



This time batch runs have been selected as the method for collecting results. This means that a batch of results will be collected before any analysis takes place. If refinement is needed, then another batch of results will be collected. The default size of the batch is usually set to 14.

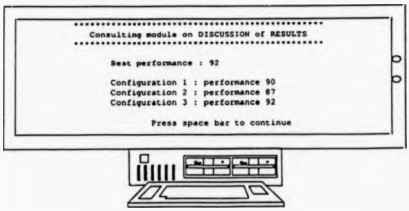


After the first run, results are collected before returning to the experiment when WES will display:

Notice that the seed value is automatically increased to randomise the stochastic elements of the model.

After all the runs relating to the first experiment have been completed and analysed, the next experiment is presented.

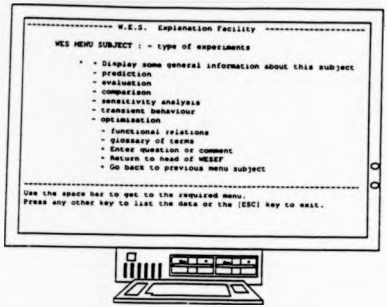
The same process as before will be repeated. When all experiments have been completed, the consultation will proceed to a discussion of the results.



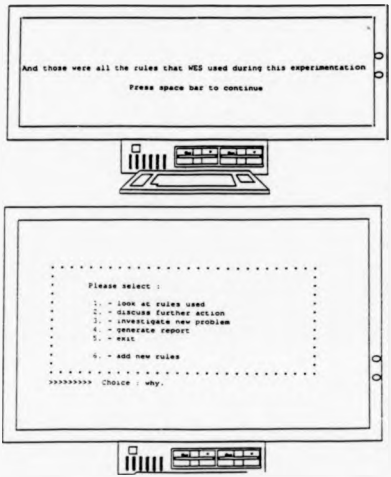
These results are presented more elegantly in the report file. They indicate however which configuration is considered best for the criteria chosen.

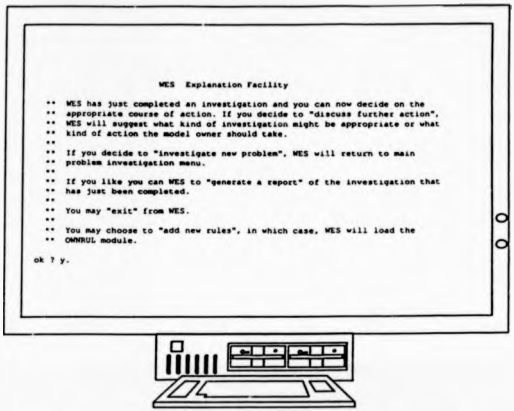

Please select :
1. - look at rules used
2. - discuss further action
3. - investigate new problem
4. - generate report
5. - exit
6. - add new rules
>>>>>>>>> Choice : 1.


WES used the following rules :
run_in_rule_4
run_time_rule_2
run_number_rule 3
prediction_rule 4
evaluation_rule_2
comparison_rule_1
comparison_rule_3
comparison_rule_2
Explain any rules ? y.
Explain run_number_rule 3 ? n.
Explain run_in_rule_4 ? n.
Explain run_time_rule_2 ? n.
Explain run_number_rule 3 ? n.
Explain prediction_rule 4 ? n.
Explain evaluation_rule_2 ? n.
Explain comparison_rule_1 ? n.
Explain comparison_rule_3 ? n.
Explain comparison_rule_2 ? y.
comparison_rule_(2).
This rule was used because WES had determined that all the configurations
that had to be compared had been compared, and therefore no more
experimentation was required before a discussion of results could begin.
ok ? n.



The user is assumed to not want to pursue this inquiry any further and presses the [ESC] key which returns him to the consultation.





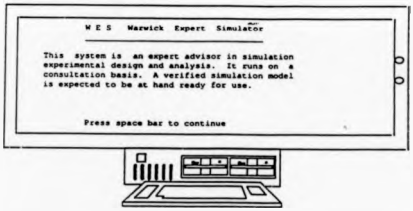
The user is now assumed to be satisfied.

Appendix 13 - c : Consultation 3 using a service station simulation model

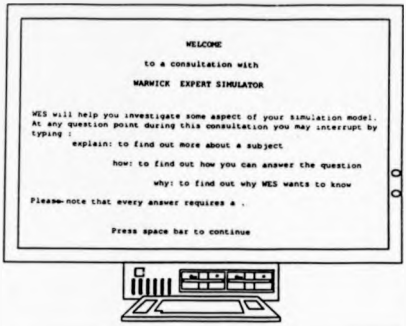
In this appendix, the following are uncovered:

- the transient behaviour task
- the stand alone version of WES
- the use of existing file about the model


WWW WWW EEEEEEEEEEE SSSSSSSSSS
WWW WWW EEEEEEEEEEE SSSSSSSSSS
WW WW EE SS
WW WW EE SS
WW W WW EEEEEEE SSSSSSSSSS
WWW W WWW EE SS
WW WWW WW EE SS
WWW WWW EEEEEEEEEEE SSSSSSSSSS
WW WW EEEEEEEEEEE SSSSSSSSSS
Developed by
Warwick Expert Simulation Centre
Version 1.0 April 1987
Press space bar to continue


W E S Warwick Expert Simulator
This system is an expert advisor in simulation
experimental design and analysis. It runs on a
consultation basis. A verified simulation model
is expected to be at hand ready for use.
Press space bar to continue

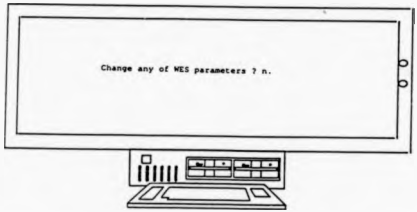

Warwick Expert Simulator
Please select :
* Consultation - stand alone
- linked to MICRO-VISION models
- linked to WITNESS models
Explanation Facility
Own Rules Development Mechanism
Comment or Question Input
Quit
Press space bar to move the cursor
Press (return) to select option



WES has learnt that the consultation will deal with a model for which it already possess a file of information. This is retrieved the moment the user specifies that it is not a new model.

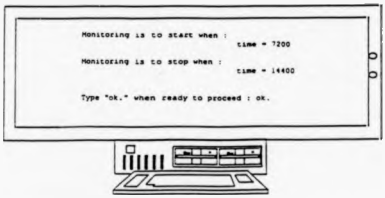
In this case, the user forgot the exact name of the entity.

This is the simplest of transient behaviour exercises.

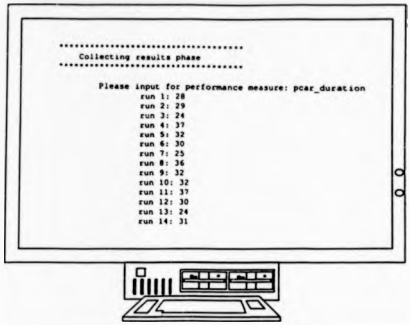


Again WES proceeds through the same reasoning pattern. It does not need to ask about the frame to be used nor the mode of result collection since these facts would have asserted in the previous consultation.

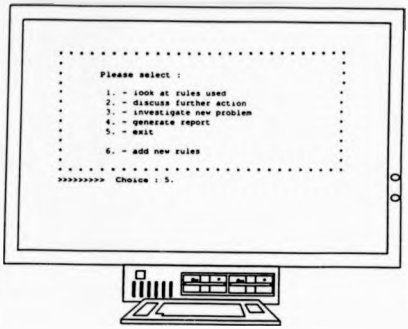
In the use of transient behaviour experiments, WES must declare to the use the extra design characteristics of the experiment.



The user is told how to monitor his experiment.



Because typing in all the results is so tedious, a stand alone experiment only retrieves the results that are essential to the problem under investigation. This naturally limits the system's ability to retrieve past data.

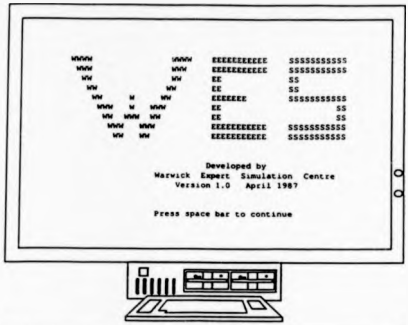


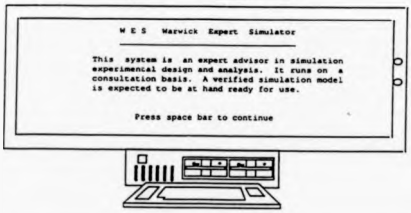
The consultation ends.

Appendix 13 - d : Consultation 4 with a municipal dump simulation model

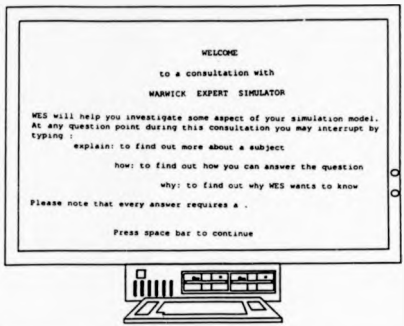
This appendix presents :

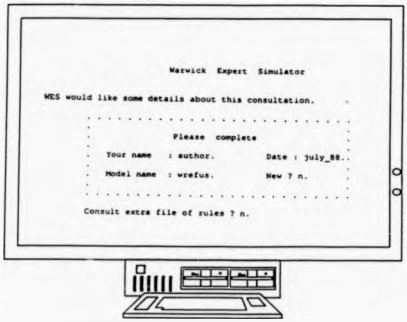
- the sensitivity analysis task.


WWWW WWWW EEEEEEEEEEE SSSSSSSSSS
WWW WWW EEEEEEEEEEE SSSSSSSSSS
WW WW EE SS
WW WW EE SS
WW W WW EEEEEEE SSSSSSSSSS
WWW W WWW EE SS
WW WWW WW EE SS
WWW WWW EEEEEEEEEEE SSSSSSSSSS
WW WW EEEEEEEEEEE SSSSSSSSSS
Developed by
Warwick Expert Simulation Centre
Version 1.0 April 1987
Press space bar to continue

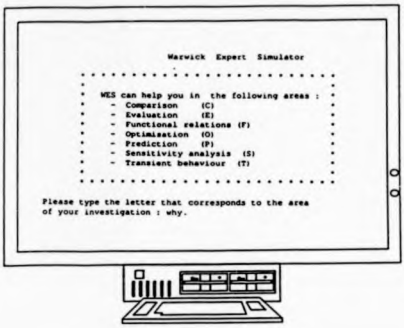

W E S Warwick Expert Simulator
This system is an expert advisor in simulation experimental design and analysis. It runs on a consultation basis. A verified simulation model is expected to be at hand ready for use.
Press space bar to continue


Warwick Expert Simulator
Please select :
Consultation - stand alone
- linked to MICRO-VISION models
- linked to WITNESS models
Explanation Facility
Own Rules Development Mechanism
Comment or Question Input
Quit
Press space bar to move the cursor
Press [return] to select option



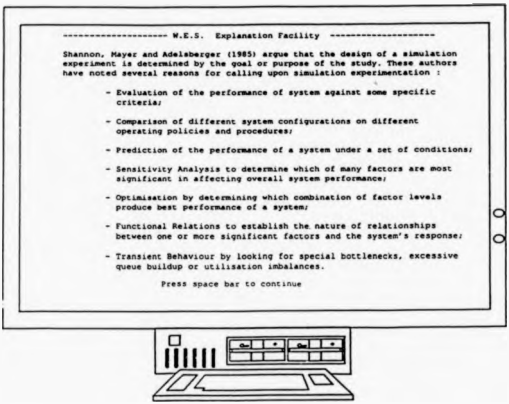


Again, WES is told that the consultation is to deal with a model for which it already posses a file of information.


Warwick Expert Simulator
WES can help you in the following areas :
- Comparison (C)
- Evaluation (E)
- Functional relations (F)
- Optimisation (O)
- Prediction (P)
- Sensitivity analysis (S)
- Transient behaviour (T)
Please type the letter that corresponds to the area
of your investigation : why.

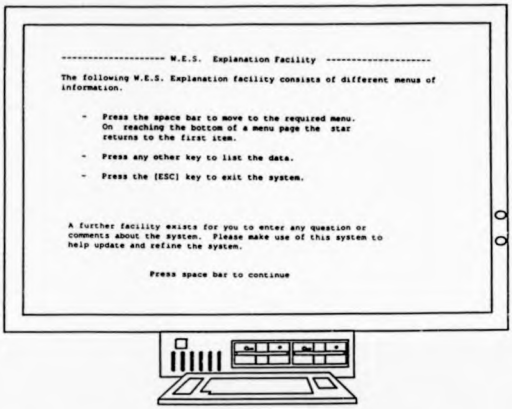

WES Explanation Facility
** Simulation is a tool that can be used to investigate a number of real
** life problems. These problems can be divided into categories :
** comparison, evaluation, functional relations, optimisation, prediction,
** sensitivity analysis, transient behaviour .
**
** By typing in a letter, you will be helping WES understand your problem
** much faster. WES will home in on the characteristics of your problem in
** a more systematic manner.
**
** In this case, since this is an "evaluation study", you should type in
** the letter "e" followed by a full stop.
OK ? n.

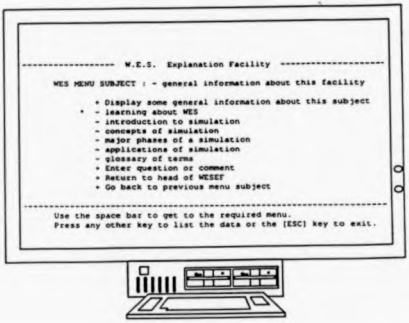

---------- W.E.S. Explanation Facility ----------
WES MENU SUBJECT : - type of experiments
* + Display some general information about this subject
- prediction
- evaluation
- comparison
- sensitivity analysis
- transient behaviour
- optimisation
- functional relations
- glossary of terms
+ Enter question or comment
+ Return to head of WESEF
+ Go back to previous menu subject
Use the space bar to get to the required menu.
Press any other key to list the data or the [ESC] key to exit.*

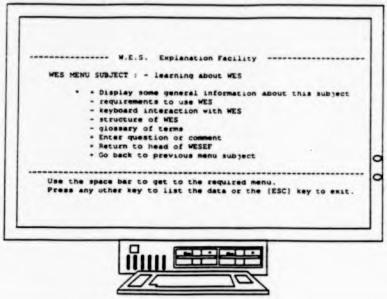

---------- W.E.S. Explanation Facility ----------
Shannon, Mayer and Adelsberger (1985) argue that the design of a simulation experiment is determined by the goal or purpose of the study. These authors have noted several reasons for calling upon simulation experimentation :
- Evaluation of the performance of system against some specific criteria;
- Comparison of different system configurations on different operating policies and procedures;
- Prediction of the performance of a system under a set of conditions;
- Sensitivity Analysis to determine which of many factors are most significant in affecting overall system performance;
- Optimisation by determining which combination of factor levels produce best performance of a system;
- Functional Relations to establish the nature of relationships between one or more significant factors and the system's response;
- Transient Behaviour by looking for special bottlenecks, excessive queue buildup or utilisation imbalances.
Press space bar to continue

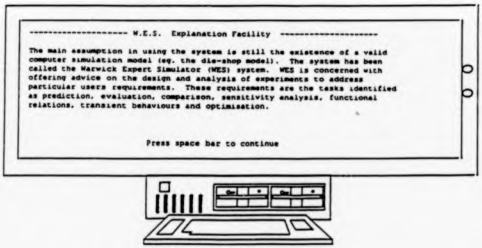

W.E.S. Explanation Facility
WES MENU SUBJECT : - type of experiments
+ Display some general information about this subject
- prediction
- evaluation
- comparison
- sensitivity analysis
- transient behaviour
- optimisation
- functional relations
- glossary of terms
+ Enter question or comment
+ Return to head of WESEF
+ Go back to previous menu subject
Use the space bar to get to the required menu.
Press any other key to list the data or the [ESC] key to exit.

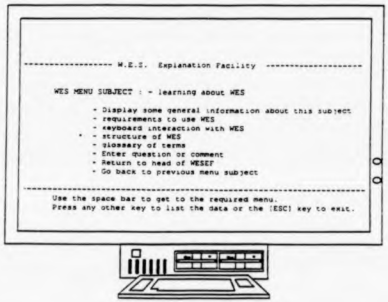

W.E.S. Explanation Facility
A WES facility developed by
Warwick Expert Simulation Centre
Version 2.1 May 1988
Press space bar to continue


---------- W.E.S. Explanation Facility ----------
The following W.E.S. Explanation facility consists of different menus of information.
- Press the space bar to move to the required menu. On reaching the bottom of a menu page the star returns to the first item.
- Press any other key to list the data.
- Press the [ESC] key to exit the system.
A further facility exists for you to enter any question or comments about the system. Please make use of this system to help update and refine the system.
Press space bar to continue

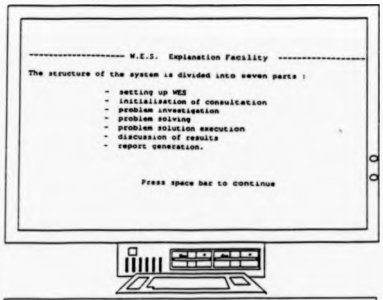

---------- W.E.S. Explanation Facility ----------
WES MENU SUBJECT : - general information about this facility
+ Display some general information about this subject
* - learning about WES
- introduction to simulation
- concepts of simulation
- major phases of a simulation
- applications of simulation
- glossary of terms
+ Enter question or comment
+ Return to head of WESEF
+ Go back to previous menu subject
Use the space bar to get to the required menu.
Press any other key to list the data or the [ESC] key to exit.

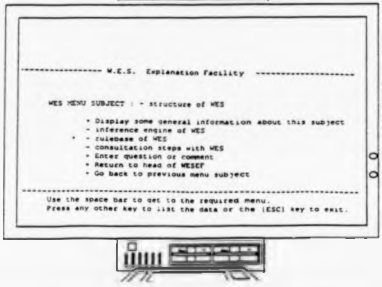

---------- W.E.S. Explanation Facility ----------
WES MENU SUBJECT : - learning about WES
* + Display some general information about this subject
- requirements to use WES
- keyboard interaction with WES
- structure of WES
- glossary of terms
+ Enter question or comment
+ Return to head of WESEF
+ Go back to previous menu subject
Use the space bar to get to the required menu.
Press any other key to list the data or the [ESC] key to exit.

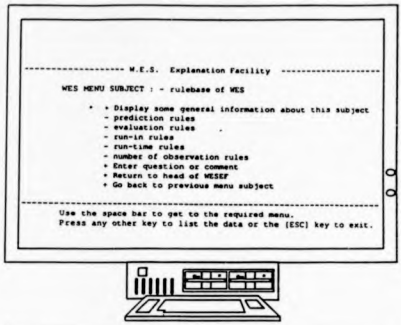

---------- W.E.S. Explanation Facility ----------
The main assumption in using the system is still the existence of a valid computer simulation model (eg. the die-shop model). The system has been called the Warwick Expert Simulator (WES) system. WES is concerned with offering advice on the design and analysis of experiments to address particular users requirements. These requirements are the tasks identified as prediction, evaluation, comparison, sensitivity analysis, functional relations, transient behaviours and optimisation.
Press space bar to continue

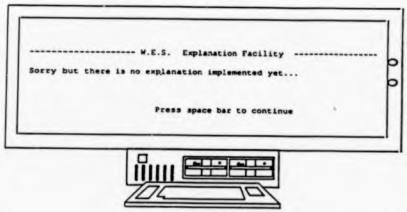

---------- W.E.S. Explanation Facility ----------
WES MENU SUBJECT : - learning about WES
+ Display some general information about this subject
- requirements to use WES
- keyboard interaction with WES
* + structure of WES
- glossary of terms
+ Enter question or comment
+ Return to head of WESEF
+ Go back to previous menu subject
Use the space bar to get to the required menu.
Press any other key to list the data or the [ESC] key to exit.

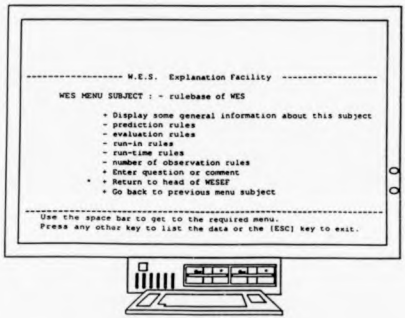

---------- W.E.S. Explanation Facility ----------
WES MENU SUBJECT : - structure of WES
* - Display some general information about this subject
- inference engine of WES
- rulebase of WES
- consultation steps with WES
- Enter question or comment
- Return to head of WESEF
- Go back to previous menu subject
Use the space bar to get to the required menu.
Press any other key to list the data or the (ESC) key to exit.

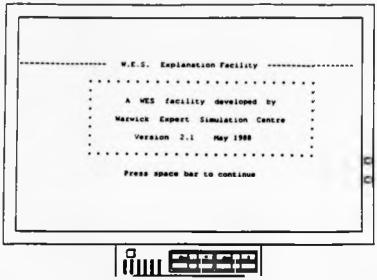

---------- W.E.S. Explanation Facility ----------
The structure of the system is divided into seven parts :
- setting up WES
- initialisation of consultation
- problem investigation
- problem solving
- problem solution execution
- discussion of results
- report generation.
Press space bar to continue

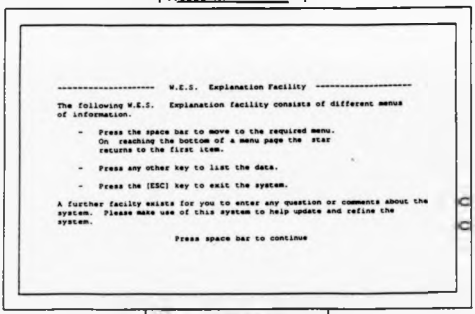

---------- W.E.S. Explanation Facility ----------
WES MENU SUBJECT : - structure of WES
- Display some general information about this subject
- inference engine of WES
* - rulebase of WES
- consultation steps with WES
- Enter question or comment
- Return to head of WESEF
- Go back to previous menu subject
Use the space bar to get to the required menu.
Press any other key to list the data or the (ESC) key to exit.

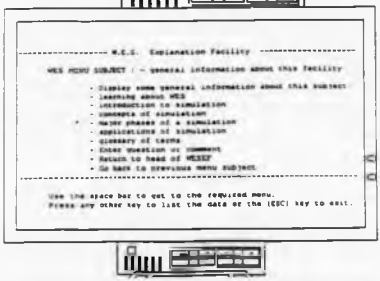

W.E.S. Explanation Facility
WES MENU SUBJECT : - rulebase of WES
* + Display some general information about this subject
- prediction rules
- evaluation rules
- run-in rules
- run-time rules
- number of observation rules
+ Enter question or comment
+ Return to head of WESEF
+ Go back to previous menu subject
Use the space bar to get to the required menu.
Press any other key to list the data or the [ESC] key to exit.


W.E.S. Explanation Facility
Sorry but there is no explanation implemented yet...
Press space bar to continue

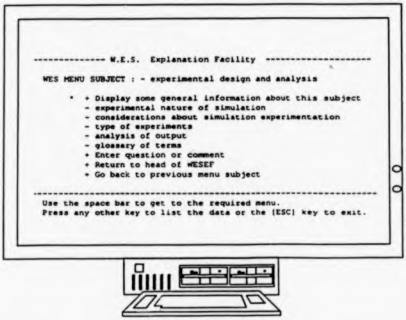

W.E.S. Explanation Facility
WES MENU SUBJECT : - rulebase of WES
+ Display some general information about this subject
- prediction rules
- evaluation rules
- run-in rules
- run-time rules
- number of observation rules
+ Enter question or comment
* + Return to head of WESEF
+ Go back to previous menu subject
Use the space bar to get to the required menu.
Press any other key to list the data or the [ESC] key to exit.

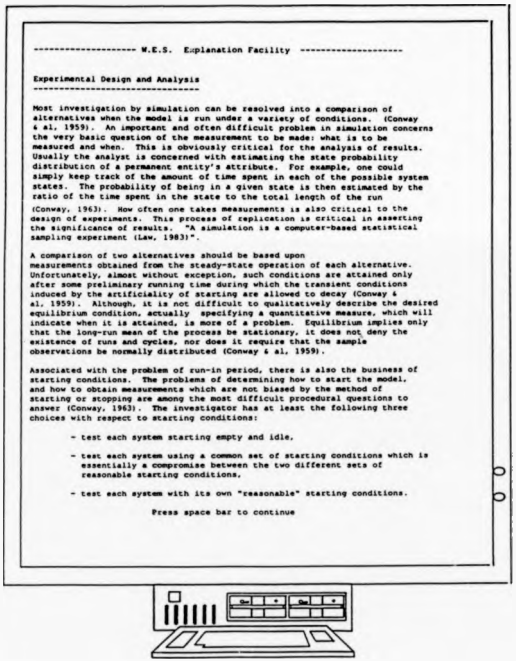

W.E.S. Explanation Facility
A WES facility developed by
Warwick Expert Simulation Centre
Version 2.1 May 1988
Press space bar to continue


W.E.S. Explanation Facility
The following W.E.S. Explanation facility consists of different menus of information.
- Press the space bar to move to the required menu. On reaching the bottom of a menu page the star returns to the first item.
- Press any other key to list the data.
- Press the [ESC] key to exit the system.
A further facility exists for you to enter any question or comments about the system. Please make use of this system to help update and refine the system.
Press space bar to continue


W.E.S. Explanation Facility
WES MENU SUBJECT : - general information about this facility
- Display some general information about this subject
- learning about WES
- introduction to simulation
- concepts of simulation
- major phases of a simulation
- applications of simulation
- glossary of terms
- Enter question or comment
- Return to head of WESEF
- Go back to previous menu subject
Use the space bar to get to the required menu.
Press any other key to list the data or the [ESC] key to exit.


---------- W.E.S. Explanation Facility ----------
WES MENU SUBJECT : - major phases of a simulation
+ Display some general information about this subject
- formulation
- programming
- verification
- validation
* - experimental design and analysis
- implementation
- glossary of terms
+ Enter question or comment
+ Return to head of WESEF
+ Go back to previous menu subject
Use the space bar to get to the required menu.
Press any other key to list the data or the (ESC) key to exit.


---------- W.E.S. Explanation Facility ----------
WES MENU SUBJECT : - experimental design and analysis
* + Display some general information about this subject
- experimental nature of simulation
- considerations about simulation experimentation
- type of experiments
- analysis of output
- glossary of terms
+ Enter question or comment
+ Return to head of WESEF
+ Go back to previous menu subject
Use the space bar to get to the required menu.
Press any other key to list the data or the (ESC) key to exit.



---------- W.E.S. Explanation Facility ----------

Experimental Design and Analysis

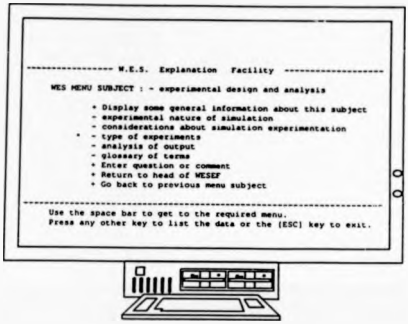
Most investigation by simulation can be resolved into a comparison of alternatives when the model is run under a variety of conditions. (Conway & al, 1959). An important and often difficult problem in simulation concerns the very basic question of the measurement to be made: what is to be measured and when. This is obviously critical for the analysis of results. Usually the analyst is concerned with estimating the state probability distribution of a permanent entity's attribute. For example, one could simply keep track of the amount of time spent in each of the possible system states. The probability of being in a given state is then estimated by the ratio of the time spent in the state to the total length of the run (Conway, 1963). How often one takes measurements is also critical to the design of experiments. This process of replication is critical in asserting the significance of results. "A simulation is a computer-based statistical sampling experiment (Law, 1983)".

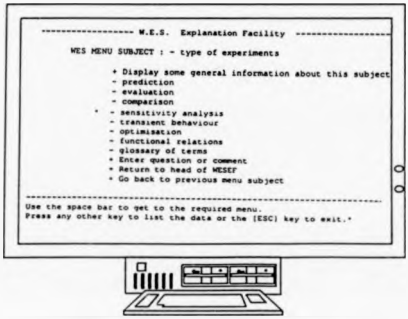
A comparison of two alternatives should be based upon measurements obtained from the steady-state operation of each alternative. Unfortunately, almost without exception, such conditions are attained only after some preliminary running time during which the transient conditions induced by the artificiality of starting are allowed to decay (Conway & al, 1959). Although, it is not difficult to qualitatively describe the desired equilibrium condition, actually specifying a quantitative measure, which will indicate when it is attained, is more of a problem. Equilibrium implies only that the long-run mean of the process be stationary, it does not deny the existence of runs and cycles, nor does it require that the sample observations be normally distributed (Conway & al, 1959).

Associated with the problem of run-in period, there is also the business of starting conditions. The problems of determining how to start the model, and how to obtain measurements which are not biased by the method of starting or stopping are among the most difficult procedural questions to answer (Conway, 1963). The investigator has at least the following three choices with respect to starting conditions:

- test each system starting empty and idle,
- test each system using a common set of starting conditions which is essentially a compromise between the two different sets of reasonable starting conditions,
- test each system with its own "reasonable" starting conditions.

Press space bar to continue

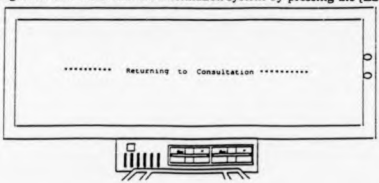

---------- W.E.S. Explanation Facility ----------
WES MENU SUBJECT : - experimental design and analysis
+ Display some general information about this subject
- experimental nature of simulation
- considerations about simulation experimentation
* - type of experiments
- analysis of output
- glossary of terms
+ Enter question or comment
+ Return to head of WESEF
+ Go back to previous menu subject
Use the space bar to get to the required menu.
Press any other key to list the data or the (ESC) key to exit.

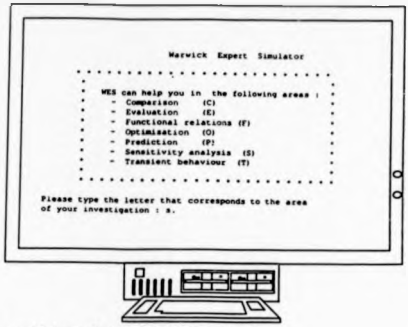

---------- W.E.S. Explanation Facility ----------
WES MENU SUBJECT : - type of experiments
+ Display some general information about this subject
- prediction
- evaluation
- comparison
* - sensitivity analysis
- transient behaviour
- optimisation
- functional relations
- glossary of terms
+ Enter question or comment
+ Return to head of WESEF
+ Go back to previous menu subject
Use the space bar to get to the required menu.
Press any other key to list the data or the (ESC) key to exit.*


---------- W.E.S. Explanation Facility ----------
Sensitivity Analysis
This expert task extends the previous task of comparison. Once a general criterion (be it a cost function) has been identified as a means of assessing performance of a system, WES can carry out some sensitivity analysis. The task of sensitivity analysis is limited to varying all the parameters by a small amount and note the change in performance. This task offers the chance of implementing the notion of strategic reasoning, where the system reacts to results from previous experiments. The main idea is to start with a default configuration and change each parameter in turn by a small positive increase and a small negative increase. Experiments are run in order to decide which of many factors are most significant in affecting overall system performance.
Press space bar to continue

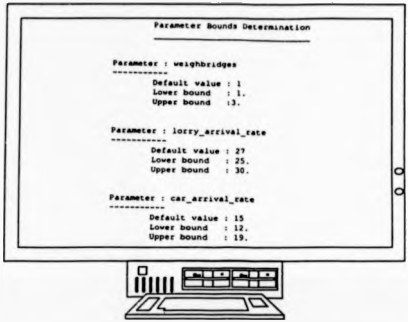

---------- W.E.S. Explanation Facility ----------
WES MENU SUBJECT : - type of experiments
+ Display some general information about this subject
- prediction
- evaluation
- comparison
- sensitivity analysis
* - transient behaviour
- optimisation
- functional relations
- glossary of terms
+ Enter question or comment
+ Return to head of WESEF
+ Go back to previous menu subject
Use the space bar to get to the required menu.
Press any other key to list the data or the [ESC] key to exit.

At this stage, the user exits from the information system by pressing the [ESC] key

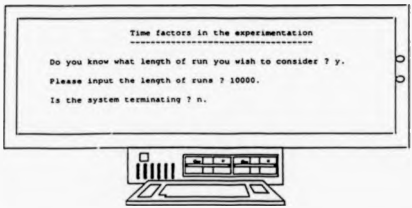




The sensitivity analysis task is finally selected.

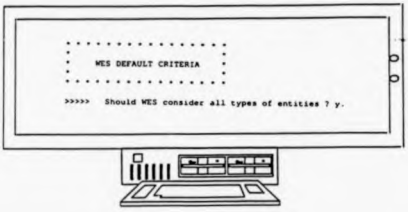


These operations of parameter determination only take place once. Obviously, this had not happened in the previous consultation. WES therefore asked the user to input the lower and upper bounds on the parameter values. (The default values would have been known).

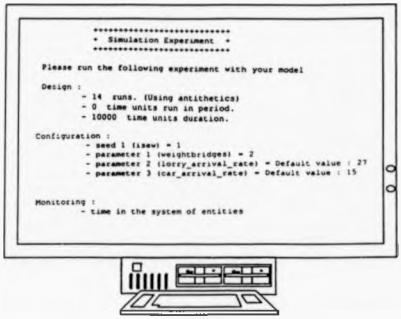


In this example, the user of WES may override any sensible notion of time in the experiment by inputing his own values as in this example.

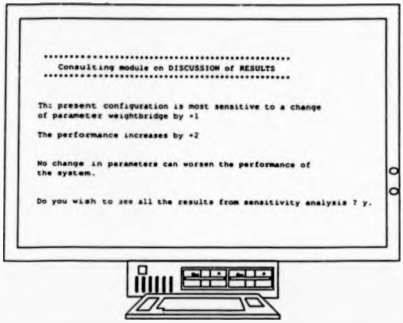
The user is concerned with minimising the time in the system of entities.

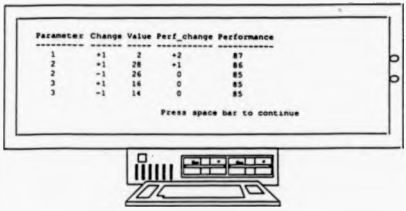


Results would then be collected, and the next thirteen runs would be performed before the results are analysed for this configuration. The next experiment would involve a change in the first parameter.



This process will be repeated for as many experiment as WES devises. In this example, WES will have considered 6 configurations in total.

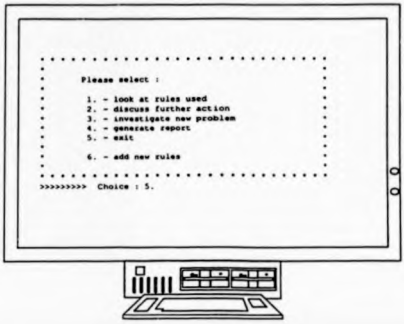




This example reflects some of the difficulty of finding a suitable evaluation criteria measure. The results suggest that the system is incentive to a change in the arrival rate of cars to the rubbish dump. This is not so. The difference are best seen by the actual processed results from which the criteria is derived. The difference in time in the system of the different elements are for each experiment :

155(default), 137, 144, 158, 150, 157

It can be seen that there is the expected progression in performance : the criteria unfortunately loses much of this information.



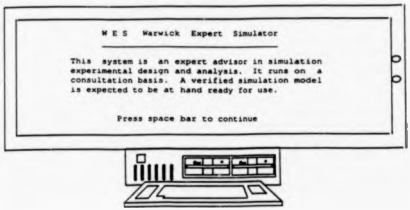
The user quits the consultation.

Appendix 13 - e : Consultation 5 using a airside-activity simulation model

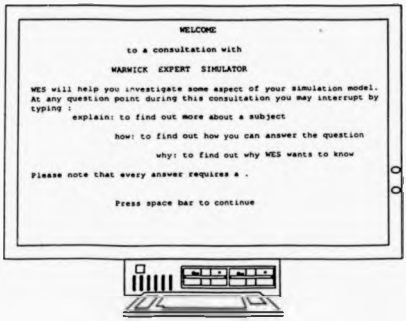
This appendix demonstrates the use of :

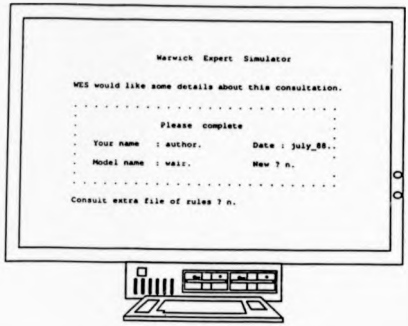
- the functional relations task;

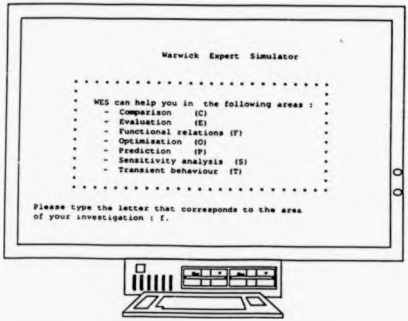
- the possibility to modify the default values of WES heuristics.

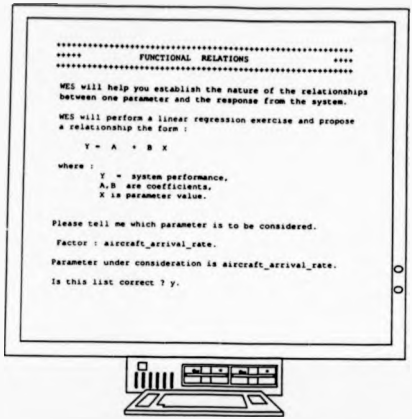



Warwick Expert Simulator
Please select :
Consultation - stand alone
- linked to MICRO-VISION models
- linked to WITNESS models
Explanation Facility
Own Rules Development Mechanism
Comment or Question Input
Quit
Press space bar to move the cursor
Press [return] to select option

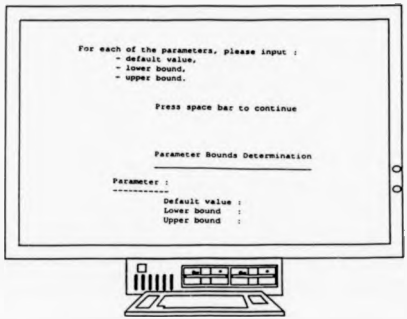

WELCOME
to a consultation with
WARWICK EXPERT SIMULATOR
WES will help you investigate some aspect of your simulation model.
At any question point during this consultation you may interrupt by typing :
explain: to find out more about a subject
how: to find out how you can answer the question
why: to find out why WES wants to know
Please note that every answer requires a .
Press space bar to continue


Warwick Expert Simulator
WES would like some details about this consultation.
Please complete
Your name : author.
Date : july_88..
Model name : wair.
New ? n.
Consult extra file of rules ? n.


Warwick Expert Simulator
WES can help you in the following areas :
- Comparison (C)
- Evaluation (E)
- Functional relations (F)
- Optimisation (O)
- Prediction (P)
- Sensitivity analysis (S)
- Transient behaviour (T)
Please type the letter that corresponds to the area
of your investigation : f.



The introduction to this functional relation task is the old one as explained in chapter 4. WES will not actually perform the linear regression exercise but only present some data points that can be used for such an exercise.



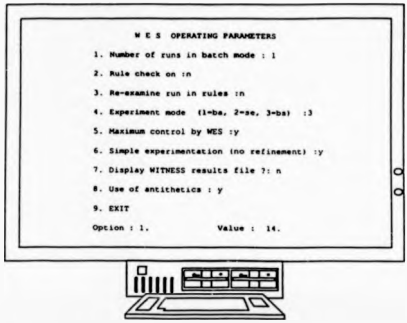
In this example, WES already knew all the values for the parameters and in fact no question is

actually asked

The user's answers will force WES to undertake some search for a suitable run in time.

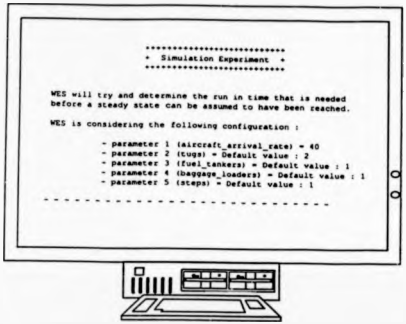
In this example consultation, WES did not ask the user about the experimental mode to use for result collection. As he cannot remember if it is set suitably, he may request look at the default

values.



The user notices that the number of runs in batch mode had been set to one. He changes this by noting option 1 to the value 14. After this, the revised table is presented.

Since no more changes are required, the user exits from this table. The user of this table of WES parameter variables permits high level consultation for an expert user. The user requiring help need never change any of theses values.



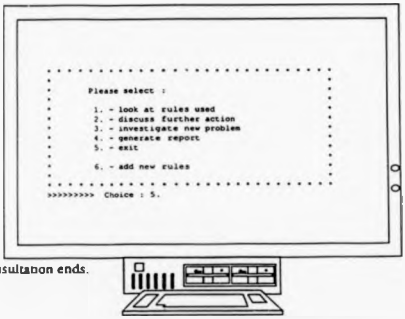
First experiment is to determine a suitable run-in period for the first configuration considered.

After the first experiment is complete, WES considers a second configuration.

WES needs to establish a suitable run in time for this configuration. In this case in happens to be different to the previous configuration.

The process is repeated for all configurations that WES considers

The table goes on with a series of paired values that may be used for some regression analysis exercise.

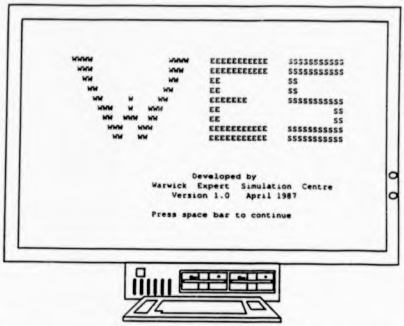


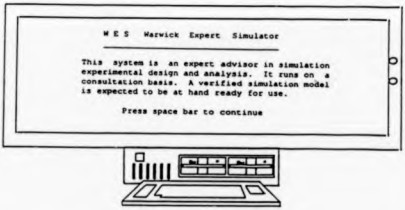
The consultation ends.

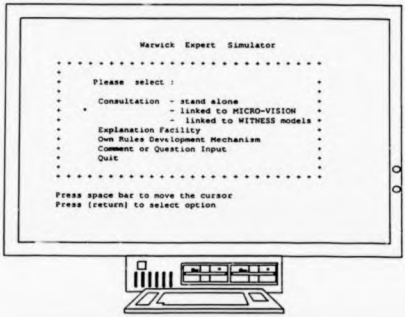
Appendix 13 - f : Consultation 6 using an immigration office simulation model

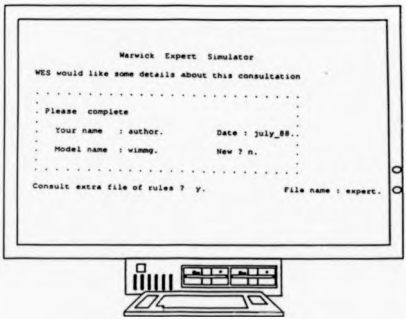
In this appendix, the following features are demonstrated:

- the optimisation task
- the use of extra rule file using uncertainty factors

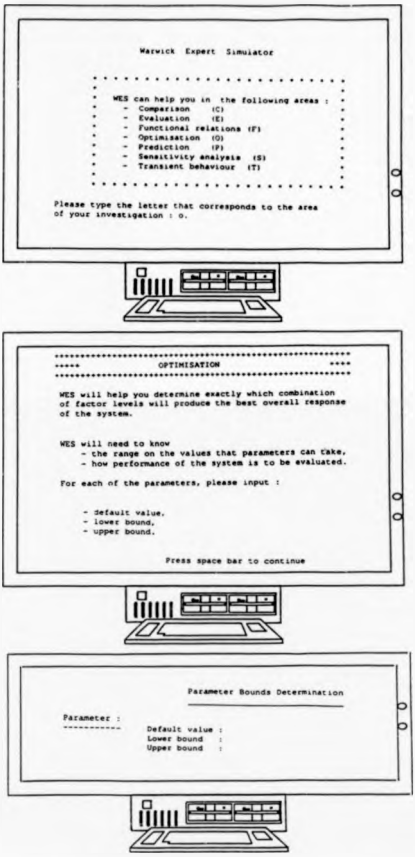

WWWW WWWW EEEEEEEEEEE SSSSSSSSSS
WWW WWW EEEEEEEEEEE SSSSSSSSSS
WW WW EE SS
WW WW EE SS
WW W WW EEEEEEE SSSSSSSSSS
WW W WW EE SS
WW WWW WW EE SS
WWW WWW EEEEEEEEEEE SSSSSSSSSS
WW WW EEEEEEEEEEE SSSSSSSSSS
Developed by
Warwick Expert Simulation Centre
Version 1.0 April 1987
Press space bar to continue


W E S Warwick Expert Simulator
This system is an expert advisor in simulation experimental design and analysis. It runs on a consultation basis. A verified simulation model is expected to be at hand ready for use.
Press space bar to continue


Warwick Expert Simulator
Please select :
Consultation - stand alone
- linked to MICRO-VISION
- linked to WITNESS models
Explanation Facility
Own Rules Development Mechanism
Comment or Question Input
Quit
Press space bar to move the cursor
Press [return] to select option

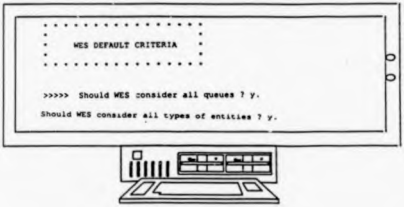


The user has expressed the desire to use the rules developed in the file called 'expert'



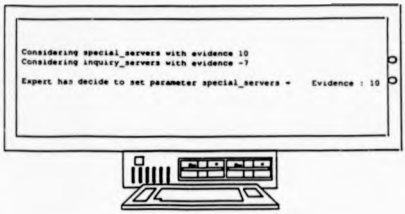
There is no actual question asked since WES actually knows all the necessary parameter values.


Time factors in the experimentation
Do you know what length of run you wish to consider ? n.
Is the system terminating ? n.
EVALUATION CRITERIA MECHANISM
Please select the type of criteria you require for evaluation purposes :
1.- WES default criteria (weighted).
2.- Your general criteria.
3.- Minimise time in system (weighted).
4.- Maximise system utilisation (weighted).
5.- Cost function.
6.- WES default criteria (absolute).
7.- Minimise time in system (absolute).
8.- Maximise system utilisation (absolute).
9.- Do not know...
>>>> Please input your choice : 6.


WES DEFAULT CRITERIA
>>>>> Should WES consider all queues ? y.
Should WES consider all types of entities ? y.

Using 'expert' rules has managed to discover the relations between the parameter of the model, the services and entity types.

Experimentation may proceed. First, WES will run the default configuration according to suitable design. Once results have been collected and analysed for this configuration, the consultation proceeds as follows.



The user is presented with the decision that WES is taking by using the expert set of rules.

Experimentation proceeds in this way. When all results have been collected and analysed, conclusions are presented.

End of consultation.

TITLE An Artificial Intelligence Framework for Experimental Design and Analysis in Discrete Event Simulation

AUTHOR *Richard Paul TAYLOR*

INSTITUTION and DATE UNIVERSITY OF WARWICK 1988