ISBN 0-315-54803-7

Canada

KNOWLEDGE ACQUISITION
IN WATER RESOURCES ENGINEERING

BY

KIM BARLISHEN

A Thesis
Submitted to the Faculty of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

Department of Civil Engineering
University of Manitoba
Winnipeg, Manitoba

August, 1989

KNOWLEDGE ACQUISITION IN WATER RESOURCES ENGINEERING

BY

KIM BARLISHEN

A thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

ABSTRACT

Expert system projects are receiving widespread attention. The major task in these projects is the acquisition of knowledge from a human expert for transfer to a computer system. This thesis examines the crucial task of knowledge acquisition, for engineering expert systems. The main objective was to investigate expert system development within existing water resources engineering problem solving environments.

In order to accomplish this objective, an extensive literature review was conducted. More importantly, the work and results of an expert systems project in the reservoir management domain are presented. Prototype expert systems were developed to accompany a standard computer model used by an experienced engineer at Manitoba Hydro. As well, to contrast this complex undertaking, the development of a prototype expert system in the preliminary dam design domain is described.

As a result of this research, guidelines are presented with regards to knowledge acquisition and the development of expert systems for water resources engineering applications. Furthermore, it is suggested that expertise is shown, in this domain, through the combined use of data, theoretical knowledge and operational experience. The relative amounts of these problem solving components affects the ease of knowledge transfer to an expert system.

ACKNOWLEDGEMENTS

This research was possible only through the combined efforts of many people. I wish to express my sincerest appreciation to Dr. D. Burn and Dr. S. Simonovic, my co-supervisors, for their encouragement and guidance during this study. In addition, I would like to thank Mr. A.D. Cormie and Mr. P.E. Barritt-Flatt for their valuable assistance, and time spent, in connection with this research. A further thanks to Mr. A.D. Cormie for acting as the 'guinea pig' in our expert systems project.

I am also indebted to Mr. Rick Welwood, from Manitoba Hydro, who provided much needed, and appreciated, computer-related advice.

I would like to acknowledge the financial support provided by Manitoba Hydro, through its Research and Development Committee, and the Natural Sciences and Engineering Research Council of Canada.

A special thanks to my colleagues from FIDS, who provided technical help, but, more importantly, daily entertainment. I would especially like to thank Andrew Nagy, my research partner on the Manitoba Hydro project. I learned a great deal through our discussions and our project work.

Finally, I would like to thank my weird, but wonderful, family and friends, who gave me support and encouragement, but also took my mind off of this research when it needed to be.

TABLE OF CONTENTS

ABSTRACT	**ii**
ACKNOWLEDGEMENTS	**iii**
TABLE OF CONTENTS	**iv**
LIST OF TABLES	**vii**
LIST OF FIGURES	**viii**
Chapter	page
1. INTRODUCTION	**1**
1.1 Problem	1
1.2 Purpose	2
1.3 Scope	3
1.4 Literature review	4
2. EXPERT SYSTEMS	**9**
2.1 Basic components of expert systems	9
2.2 Knowledge representation in expert systems	12
2.3 Expert system development	14
2.3.1 Tools	14
2.3.2 Project participants	16
2.3.3 Development life-cycle	17
3. ENGINEERING EXPERTISE AND EXPERT SYSTEMS	**20**
3.1 Water resources engineering	20
3.2 Expert systems as "intelligent" interfaces	22
4. TRANSFERRING KNOWLEDGE	**26**
4.1 Approaches	26

Chapter page

4.2 Knowledge engineering 29

4.3 Selection of a knowledge engineer 31

4.4 Standard training process 32

5. MANITOBA HYDRO CASE STUDY 35

5.1 Problem related objectives 35

5.2 The Reservoir and Energy Scheduling Engineer 35

5.3 The EMMA computer model 37

5.4 Research objectives 43

6. KNOWLEDGE ACQUISITION PROCEDURE 44

6.1 Learning about the domain 44

6.2 Constraining the problem 46

6.3 Knowledge acquisition sessions 48

6.4 Lessons learned in knowledge acquisition 49

6.5 Testing 53

6.6 Importance of the prototype to development 54

6.7 Limitations of expert systems 55

7. PROTOTYPE SYSTEMS 57

7.1 Manitoba Hydro prototype systems 58

7.1.1 Description of the pre-processor 59

7.1.2 Description of the post-processor 64

7.1.3 Current state of the prototype systems 73

7.2 Preliminary dam design case study 74

7.2.1 Description of the problem domain 74

7.2.2 Development of a prototype system 75

Chapter page

8. ENGINEERING KNOWLEDGE 80

8.1 Engineering today 80

8.2 The computer - friend or foe? 81

8.3 Possible roles of expert systems 84

8.4 Classifying water resources engineering expertise 85

8.4.1 According to problem type 85

8.4.2 According to knowledge type 88

8.4.3 According to knowledge structure and origin 91

8.4.3.1 Data analysis 91

8.4.3.2 Mathematical modelling knowledge 93

8.4.3.3 Operational experience 95

8.5 Rule development 98

9. RECOMMENDATIONS 102

9.1 Water resources expertise 102

9.2 Expert systems in water resources engineering 105

9.3 For the development team 107

9.4 For the knowledge engineer 109

10. CONCLUSIONS 111

REFERENCES 114

Appendix page

A. Listing of Pre-processor to EMMA 118

B. Listing of Post-processor to EMMA 132

C. Listing of Preliminary Dam Design Prototype 162

LIST OF TABLES

page

Table 1. Size classifications for dams. 76

LIST OF FIGURES

page

Figure 1. Basic structure of an expert system. 10

Figure 2. Development life-cycle for expert systems. 18

Figure 3. Expert systems as "intelligent" interfaces. 24

Figure 4. The engineering expert system approach. 28

Figure 5. Hierarchical model of problem formulation in EMMA. 51

Figure 6. The pre-processor to EMMA. 60

Figure 7. The post-processor to EMMA. 65

CHAPTER 1.

INTRODUCTION

1.1 Problem

The newest phase in the evolution of computers in the work place, Artificial Intelligence (AI), and its goal of creating more "human-like" computer systems, has become the focus of much research and speculation. What impact might this trend have on the field of water resources engineering? One particularly successful branch of AI looks at the representation and manipulation of human problem solving knowledge within a computer system. Slowly, these knowledge-based systems are making their way into the realm of water resources engineering. In particular, expert systems, which try to encode the expertise of an experienced engineer, could become additional tools for managing the complex and expansive knowledge requirements of the water resources domain.

The reason for the interest in engineering applications of expert system technology is the apparent suitability of many of their problems to AI techniques. Expertise and intuitive judgement form an important aspect of civil engineering, making the development of expert systems highly relevant [1]. However, some branches of civil engineering have seen more successful applications than others, due primarily to the characteristics of the various subdomains. Knowledge-based systems are often constructed to perform tasks requiring knowledge, but not necessarily scarce human expertise. Such systems might assist an engineer in performing routine, time-consuming tasks, perhaps by

encoding design manuals or procedural codes. This type of relatively static knowledge is readily available and can usually quite easily be transferred into a computer usable format. There are relatively few situations in engineering, however, where computer-based applications of a straightforward set of rules or regulations would be useful [33].

In water resources engineering, uncertainty and complexity often preclude the creation of strict codes or guidelines. For this domain, expert systems become more appropriate, as experience, judgement and intuition become just as important as any equations, models, or texts describing the behaviour of water systems. Consequently, a suitable definition of a water resources expert system is a computer application that assists in solving complicated water resources problems by incorporating engineering knowledge and judgement, principles of systems analysis, experience, and intuition into the solution procedure [50].

For an expert system to exist and achieve the desired high performance level, knowledge must somehow be acquired from a human expert and correctly represented within the system. This thesis focuses some needed attention on the relatively unexplored area of knowledge acquisition, as it pertains specifically to the water resources engineering domain.

1.2 Purpose

The main objective of this research project was to characterize the knowledge a water resources expert possesses. As well, there is an examination of the knowledge acquisition process for developing expert systems in the water resources engineering domain and the suitability of

this type of expertise to such a transfer process. This same potential for research was identified by Fenves et al. [22], as expert system projects "provide for the first time an environment for conducting research on the practical aspects of the profession, namely, on how practitioners use, integrate and combine elements of textbook knowledge to perform practical and innovative tasks."

1.3 Scope

The discussions and conclusions presented in this text stem from ideas the author has developed based on an extensive literature review, discussions with fellow engineers, class work and a case study expert system project. The case study involves a joint project between the University of Manitoba and Manitoba Hydro, to develop expert systems within the reservoir management domain. The thesis, however, begins with a brief literature review, which is followed by a description of water resources engineering and the possible role for expert systems. Following this discussion, literature in the knowledge acquisition field is reviewed, and the Manitoba Hydro case study project is presented, including the knowledge acquisition procedure and the resulting prototype expert systems. To contrast this large scale and difficult project, work surrounding the development of a small scale expert system class project is also described. The discussion then focuses on defining the characteristics of water resources expertise. Finally, recommendations and conclusions are given regarding knowledge acquisition and expert system development, as related to water resources applications.

1.4 Literature review

There have been successful water resources related applications of expert system techniques reported in the literature. One recent study identified 13 systems, most of which were labelled as "work-in-progress" [51]. Two recent texts published by the American Society of Civil Engineers are devoted to the subject of expert systems in civil engineering, and present example applications in many branches of civil engineering, including water resources [36,39]. A more comprehensive literature review of this area was conducted by Simonovic and Barlishen [52]. From these publications, it is obvious that the basic engineering tasks of design, planning, and operations are represented in the current expert systems research in water resources.

To further illustrate this point, several of these water resources applications of expert systems will be briefly described. Special attention will be paid to references made to the knowledge acquisition procedures conducted in connection with these projects.

* REZES is an interactive, menu-driven consultation program designed as a tool for eliciting advice regarding reservoir analysis. The program contains several mathematical models (optimization and simulation) and helps a user: formulate his problem; select an appropriate model; prepare the necessary data; and interpret the results. The developers were also the experts, so formal knowledge acquisition issues were not discussed [51].

* FLOOD ADVISOR is a pilot expert system that provides advice regarding the most suitable design flood estimation technique based on

the availability of streamflow and precipitation data. This suitability may also depend on factors such as the importance of the project or consequences of failure. Neither the contents of the expert system, nor the process through which it was developed, are explained [20].

* An expert system has been developed to assist in selecting process units for upgrading small water supplies. The advice given reflects the experience of an expert water engineer, and his knowledge about technical aspects of water treatment (available methods and their requirements) and existing regulations and standards. The group of available treatment methods is reduced to the 'best' methods based on technical feasibility, suitability and cost. Again, minimal reference is made to the elicitation of knowledge from the expert water engineer [34].

* Jenkins and Jowitt [30] describe work undertaken towards the development of an expert system for controlling the operation of the activated sludge process at a wastewater treatment plant. The knowledge was acquired through discussions with plant managers, who possess knowledge about routine control actions and heuristic control measures, developed through experience. The article deals mainly with the knowledge involved in river basin management, which was helpful to the sections of this thesis related to water resources expertise. However, they too virtually ignore the actual knowledge acquisition procedure used to capture the pertinent river basin management expertise.

* WMS is an expert system, used by the Seattle Water Department, that aids the user in developing appropriate drought management policies based on drought intensity, time of year, and other system

characteristics. The user is provided with general drought potential information and specific guidance regarding system operations. The expert system integrates the linear programs, graphics and databases used to solve a drought management problem. Emphasis is placed on describing the knowledge contained in the systems, as opposed to how it was collected and organized [43].

* SID is an expert system, used by the Seattle Water Department, offering guidance for initiating water-use restrictions during droughts. Initial rules for offering guidance were developed by organizing 120 drought scenarios and their optimal water shortage response plan (determined through linear programming) according to starting week, initial conditions, subsequent streamflows, yield results, and economics. By comparing these rules to published water shortage response plan criteria, a final set of rules was established. The expert system is used to integrate several programming techniques: linear programming; database management; and computer graphics. Although limited in its extent, a discussion is provided on the acquisition of knowledge from several drought management experts. Specific details and problems are not mentioned [42].

* Two computer systems, HYSTOR and HYSIZE, were developed to: determine the optimal layout for a particular hydroelectric plant; rank alternatives according to economics; and test the sensitivity of assumed variables, such as fish release. HYSIZE is used for run-of-river type projects without storage, while HYSTOR is applied to sites with reservoir storage. The programs use supplied hydraulic, turbine, hydrologic and

economic data to explore: reservoir storage (HYSTOR only); turbine rated flow; and hydraulic friction loss. The programs are not referred to as expert systems, and, thus, knowledge acquisition issues are not dealt with at all [15].

* An expert system has been developed to act as a front end to the EPA's Storm Water Management Model (SWMM), which assists the user in estimating the hydrologic parameters for runoff estimation and building the SWMM input file. The system provides reasonable initial values for input parameters and helps select computational options related to runoff. It then uses simulated and observed hydrographs to adjust parameters, to calibrate the model. While the knowledge contained in the system is thoroughly described, the work that led up to the system rules is not discussed [3].

* In a two part article, Franck and Krauthammer [23,24] touch on most of the issues involved in the development of expert systems. The articles refer to work done in creating an expert system to assist during the field inspection of dams, by incorporating qualitative field observations and safety factor calculations and comparisons. The user is assisted in determining possible structural problems with the dam. The authors devote a section of one article [24] to knowledge acquisition, which, in their case, involved the expert acting as knowledge engineer.

This presentation of current expert systems research in water resources engineering is incomplete, as much of the research in the area remains unpublished. Simonovic [50] makes reference to several other systems, from his personal communication with the developers. It is,

however, evident from this list that knowledge acquisition, conducted in connection with any expert system project, has been virtually ignored in the literature. Most articles stress the actual knowledge contained in the system, as opposed to the procedures used to elicit it.

CHAPTER 2.

EXPERT SYSTEMS

2.1 Basic components of expert systems

Expert systems offer intelligent advice or make intelligent decisions in situations that would, otherwise, require some form of human expertise [48]. They may function as: an assistant to a domain expert; a partner to an expert; or a replacement for part of an expert's knowledge, to fill the void between an average worker and the expert [19]. An expert system is a computer model composed of the following components:

- user interface;
- explanation subsystem;
- knowledge base;
- working memory;
- inference engine;
- knowledge acquisition subsystem.

Figure 1 illustrates the basic structure of an expert system. The descriptions of these components that follows is taken mainly from Rolston [48].

* The **user interface** is responsible for requesting and translating user input, and presenting generated results to the user. An attempt should be made to create a highly interactive interface that provides the user with access to the information contained within the system.

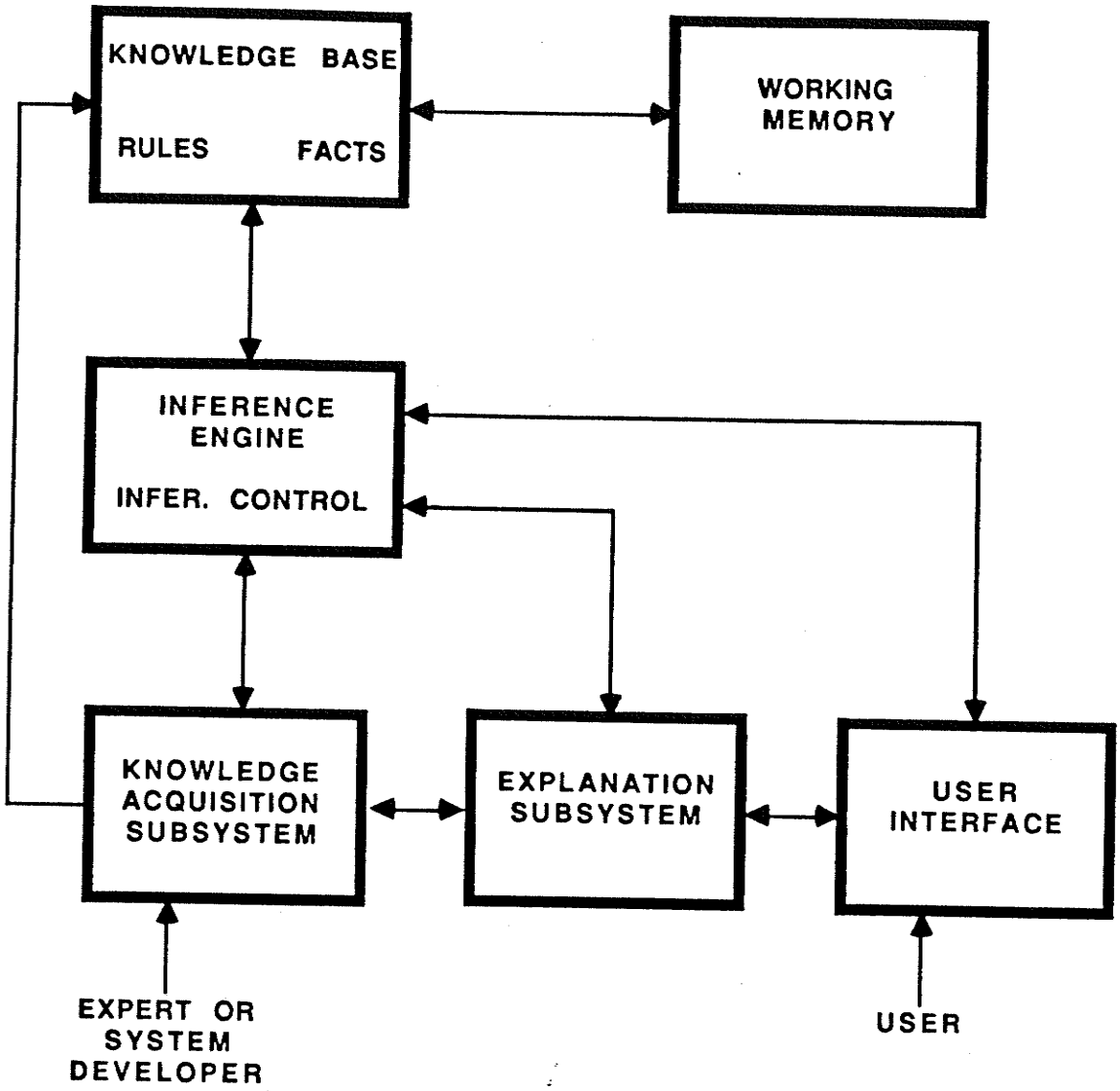

Figure 1. Basic structure of an expert system [50].

* The **explanation subsystem** is a very important part of an expert system, as it is responsible for explaining, to some extent, the reasoning behind any conclusion the system reaches. This may involve a trace of the execution of the system and/or the ability to respond to user

questions, for example, why a piece of information is needed or why a particular conclusion was not reached. The explanation facility can be important at the development stage, when the system is being debugged.

* The **knowledge base** contains the facts and rules associated with the application domain. These rules can vary from being strictly procedural (i.e., well-defined and invariant) to heuristic (practices or procedures that are valuable but are incapable of proof - "rules of thumb"), gathered through experience.

* The **working memory** contains information about the current problem to be solved. It initially contains user specified facts, but as the system reasons with the data, facts may be added, modified, or deleted. The process concludes when the desired problem solving process ends [39].

* It is the **inference engine** that controls the execution of the system, and determines how to solve a particular problem. It uses the knowledge base to modify and expand the contents of working memory [39]. Most expert systems are based on backward or forward search techniques. In backward chaining, the system begins with the desired goal of the system towards the requisite conditions to satisfy this goal. On the other hand, forward chaining uses the known conditions to work towards the desired goal of the consultation. The inference engine basically constructs algorithms to solve given problems dynamically, using these, or other, search techniques [19].

* The **knowledge acquisition subsystem** is used to perform modifications to the knowledge base. This modification process can take one of three

forms:

1) manual knowledge update - a system developer interprets information provided by a domain expert and updates the knowledge base;
2) the domain expert updates the knowledge base directly;
3) machine learning - new knowledge is generated automatically by the system based on generalizations drawn from experience.

Through its architecture, an expert system separates domain knowledge (knowledge base), problem-solving control knowledge (inference engine) and problem specific information (working memory).

2.2 Knowledge representation in expert systems

The three most common approaches to representing knowledge within an expert system will be briefly presented. Knowledge can be represented in the form of rules, semantic nets or frames. The source of this discussion is an expert systems text by Waterman [55].

Perhaps the simplest and most popular knowledge representation scheme is **rules**. This appears to be the most natural and understandable method of modelling human problem solving expertise. Rules are most appropriate when the domain knowledge results from associations between facts that have evolved through years of problem solving. Rules generally take the form of:

IF (set of conditions)
THEN (actions to be taken).

Another approach is to represent domain knowledge through a network of nodes and arcs, known as a **semantic net**. The nodes represent objects, concepts or events, and the arcs represent the relations amongst the nodes. This network structure facilitates the idea of nodes lower in a hierarchy inheriting properties from those higher up, through the arc relationships.

Finally, the term **frame** refers to a special way of representing concepts and situations. Essentially, it is the same as a semantic net, in that it consists of a system of nodes and arcs. However, all the properties of an object or concepts are collected together at a node, in a package called a frame, making it more structured than the semantic net. A node then consists of a collection of attributes related to an object, and values for these attributes. For example a node representing a dam may take the following form:

DAM FRAME

Name: Nelson Dam
Height: 50 m
Type: concrete
Purpose: water supply
Location: Nelson River
Impoundment:

Each slot can have attached procedures that are executed if information from the slot is added, removed, or needed. The emphasis is now on the nodes, rather than the links, as they contain more information.

Frames and semantic nets are most helpful when a large number of rules are involved, and grouping or structuring them is helpful. In the case studies presented, rules were found to be an appropriate form of representation. The systems developed contain relatively few rules, and

limited networks of concepts and possible conclusions, at this stage in their development. Further development on the prototype systems may reveal a need to turn to a more structured representation technique. In addition, the rule-based approach was supported by the software tools used to develop these systems.

2.3 Expert system development

2.3.1 Tools

Expert system development tools can generally be classified as either languages, shells, or environments. These tools will vary widely in their support environments (user/developer interfaces, explanation facilities, debugging aids), graphics capabilities, integration capabilities, costs, and representation of uncertainty. However, all of these factors are important considerations when selecting the most appropriate tool for a specific application. In fact, Palmer and Mar [43] feel that some of the available tools may be friendly enough to allow an expert to directly enter his knowledge. They state that the need for a knowledge engineer is eliminated, and the problem of knowledge acquisition is eased.

Although conventional **programming languages** are sometimes used, the two most common languages, for expert system development, are Lisp and PROLOG. They are, essentially, general-purpose symbol representation and manipulation languages. As languages, these tools offer great flexibility, but do not offer guidance on knowledge representation, or, in the case of Lisp, inference engines. Prolog does possess a built-in search method, making it somewhat more specific than Lisp [48].

On the other hand, expert system **shells** provide a built-in framework for more rapid development of expert systems, assuming the characteristics of the application match those offered by a particular shell. Shells usually support one of the knowledge representation schemes and inference techniques (backward or forward chaining). Often, they will also possess very friendly user and developer interfaces, which may include graphics capabilities. Most will offer an explanation facility, that can range from a simple trace of execution to developer defined or generated English-like explanations [19].

The two case studies presented within this text were developed within expert system shells. The Manitoba Hydro prototype systems were created in RuleMaster, a rule-based tool supporting forward and backward chaining. The interfaces are very friendly and the explanation facility is extensive. An important consideration in engineering applications will, no doubt, be the ability of an expert system to integrate with other software. To this end, C and FORTRAN source code generation are possible in RuleMaster. As well, rules can be directly input, or this tool has an induction algorithm allowing rules to be inferred from example tables [46].

For the design of a prototype expert system in the preliminary dam design domain, VP-Expert, a PC-based tool was utilized. It offers many powerful features, considering it is one of the lowest priced shells on the market. These features include [45]:

- simple English-like rule construction;
- rule induction from tables;
- backward chaining (forward chaining is possible);

- ability to exchange data with external files, including certain databases and spreadsheets;
- ability to execute external DOS files.

The other form expert system tools can take is that of a **development environment**, which will often incorporate multiple knowledge representation and reasoning techniques. These techniques are not necessarily restricted to the rule-based methods, and forward or backward chaining previously mentioned. Some environments will also support extensive graphics capabilities and, for efficiency, incorporate conventional programming procedures. Of course, these tools possess a very friendly user interface, and a helpful development interface. Many of these large scale tools require Lisp workstations, but they may also run on more conventional workstations. They offer more flexibility and very sophisticated capabilities, usually at a high price [19].

2.3.2 Project participants

The participants in an expert systems project will generally include [22]:

1) expert(s) - the person(s) whose domain knowledge is to be represented within the system;
2) knowledge engineer - the person responsible for acquiring, organizing, and representing the knowledge in the chosen representation scheme;
3) computer programmer - implements the acquired knowledge within the chosen development tool;

4) end user - the person who will interact with the expert system.

The system should be designed with the user's level of expertise and the intended role of the system (i.e., partner, assistant or replacement for expert) in mind. The actual number of participants depends on the available resources, the project objectives and the expected domain complexity involved.

2.3.3 Development life-cycle

The development of an expert system is essentially an iterative procedure. As shown in Figure 2 the expert system life-cycle involves the following tasks [48]:

- **problem selection** - many authors offer advice on assessing the suitability of a specific problem to solution through the expert systems approach [19,48,55];
- **prototype construction** - construction of a prototype that represents a small part of the final system;
- **formalization** - knowledge is acquired from an expert, organized and represented in a form compatible with that offered by the chosen development tool;
- **implementation** - knowledge is implemented within the chosen development tool;
- **evaluation** - the responses of the expert system are tested and any necessary modifications to the knowledge base are

made;

- **long-term evolution** - the expert system continues to evolve, perhaps by correcting or adding to the knowledge base to increase its usefulness.

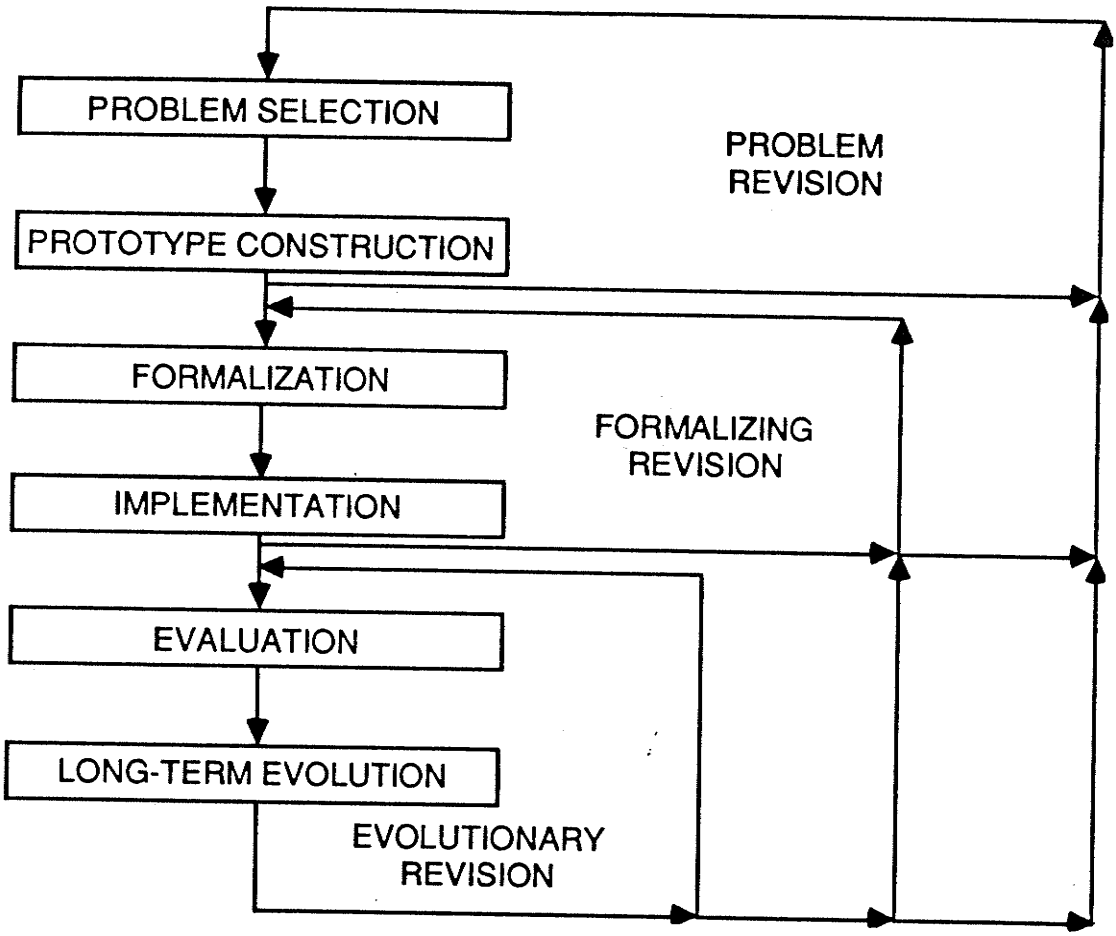

Figure 2. Development life-cycle for expert systems [48].

These descriptions will be augmented in later sections of this text dealing with the Manitoba Hydro case study expert systems project. It should,

however, be pointed out that the development of a prototype, itself, requires cycling through the formalization, implementation, and evaluation stages.

CHAPTER 3.

ENGINEERING EXPERTISE AND EXPERT SYSTEMS

3.1 Water resources engineering

It has been said that "engineering consists of acquiring and manipulating knowledge so that it can be used productively", and expert systems try to do the same thing [29]. The goal of water resources management is to ensure water is available, in sufficient quality and quantity, at the right location and time, and to protect humans and their activities from the harmful effects of water [32]. In hopes of achieving this goal, one must often combine scientific principles and computational algorithms with heuristics. In theory, engineering decisions are based on accepted underlying physical principles. Often, however, this trace-back to known facts is difficult to establish. Due to time or budgetary constraints, and incomplete or inaccurate data, the water resources engineer cannot always conduct a complete analysis of all relevant problem parameters.

Not only are fundamental principles scarce, but structured, procedural decision making is impossible. It is under these ever present circumstances that experience and judgment come into play. In contrast, much of structural engineering depends on strict design codes and procedures that have been set based on known characteristics and behaviours of many building materials. The complexities of the domain make it necessary for water resources engineers to specialize, making expertise in a particular area a scarce commodity. Since these experts possess extensive knowledge about a narrow class of problems, it becomes

possible to provide computer programs with enough knowledge to perform the same task effectively [17].

In addition, the physical systems and processes involved in water resources cannot be accurately or completely described using available techniques. An engineer is working with only a limited knowledge of the behaviour of these complex systems. Heuristics, or rules of thumb, can help compensate for a lack of complete theory and allow a problem solver to make educated guesses [10]. However, experience is needed to create heuristics that properly reflect the problem and permit more intelligent and efficient problem solving to occur. The knowledge of an experienced water resources engineer cannot, thus, be entirely represented through mathematical means.

Another complicating factor in this engineering domain is the multidisciplinary nature of water resources problems. Most decisions will have direct economic, social and environmental impacts, which are not always easy to predict or quantify. A good engineer will consider all of these factors. Such subjective, qualitative issues may not be properly dealt with by a novice engineer. A water resources engineer can no longer rely solely on a numerical analysis to justify his decision making. Increased public awareness about social and environmental concerns has led to added pressure to come up with solutions that satisfy a larger number of constraints. In the end, "engineering tasks require both the science of quantitative knowledge and the art of qualitative knowledge for establishing suitable solutions" [38].

It is hoped that expert systems can assist engineers in managing the immense and diverse knowledge requirements of complex water

resources problem domains. This requires the acquisition and representation of private human expertise, as the static knowledge found in textbooks or manuals is not enough. An engineer must have skilled and informed judgement so he can deal with probabilistic or erroneous information. This type of knowledge, the knowledge an experienced problem solver has gathered, will be organized and condensed into a highly compiled form, making it harder to analyze using introspection, but very effective for problem solving [19]. An expert problem solver can be distinguished by increased speed, reduced errors and increased adaptability [37]. As well, the knowledge base a human expert is working with is constantly maturing and expanding with each added experience. In the next section, a possible role for expert systems within the current water resources environment will be presented.

3.2 Expert systems as "intelligent" interfaces

Existing modelling tools and procedures can assist an engineer in training others in the domain, making decisions, or simply investigating the character and behaviour of a physical system. Such models can ostensibly range from singular equations or relationships to a complicated series of formulas involving a large number of problem parameters.

Several roles can be envisioned for expert systems within these current problem solving environments. An expert system can be developed as a stand-alone tool, where adequate numerical models are scarce, and problem solving relies almost exclusively on judgment and experience. It may also act as an alternative form of documentation, in place of textbooks or manuals. However, in most cases, an expert system would be

brought into an environment that already contains some form of modelling.

Due to the computer revolution and the widespread attention being paid to systems analysis techniques, many computer models have been created based on fundamental physical laws governing water systems. Engineers make extensive use of models to explore the effects of incomplete knowledge and uncertainty through data and error analyses [29]. These models must often incorporate some degree of problem simplifications or assumptions, perhaps considering only a small number of, what are deemed to be, important parameters. As well, computer models may include empirically derived relationships and, therefore, will not be applicable for all problem scenarios. It falls upon the engineer to, first of all, derive a suitable description of the problem he is dealing with, then choose an appropriate model, in light of data requirements and modelling considerations. Finally, his problem must be formulated according to the model requirements and any resulting output must be carefully analyzed and interpreted. At all stages in this procedure, engineering judgments must be made, most of which will require some degree and form of expertise.

In any domain, no matter how complete a model exists, if no one understands the modelling tool, it will sit unused. People may be intimidated by a new model if it appears too complicated to learn, or the underlying modelling techniques and assumptions are unclear. There is also always the chance a model will be improperly used. Therefore, it is suggested that a starting point for engineers interested in expert systems is looking at them as acting as cushions between a complicated

is looking at them as acting as cushions between a complicated computer tool and an inexperienced user. This approach is represented in Figure 3.

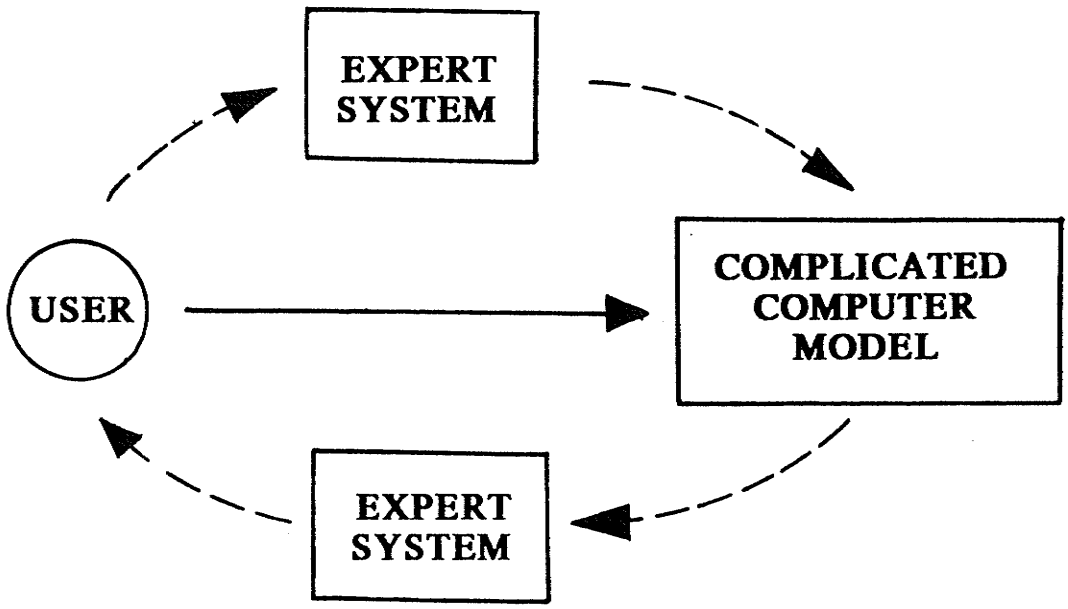

Figure 3. Expert systems as "intelligent" interfaces.

Expert systems could be developed on both the input and output sides. Acting as pre-processors or post-processors to more standard engineering software, expert systems may assist a user in judging: the suitability of this model to his specific problem; the necessary data requirements and problem formulation; the relevance and meaning of the output generated by the model; if other techniques or approaches are needed; or what other steps should be taken in the analysis [18]. A recent example of this "intelligent" interfaces approach was the development of expert systems for calibrating the EPA's Storm Water Management Model (SWMM) [3].

This approach was also applied in selecting an appropriate problem domain in which to study the possible integration of expert systems with current water resources problem solving environments. Engineers are accustomed to working with computers, and expert systems designed to accompany existing computer models may be more readily accepted. Part of the research this thesis describes centers around a complicated computer modelling tool used within Manitoba Hydro for reservoir management.

Perhaps, the extended vision of expert systems should be as one component of a hybrid decision support system. As such, expert systems can be linked together with the systems approach, data management systems, engineering expertise and computer graphics [51]. This approach to development was taken by Palmer et al. [42,44], who use expert systems to integrate databases, mathematical programming and graphics.

The following section will give an overview of the concepts surrounding the knowledge acquisition procedure used to create "intelligent" interfaces to a reservoir management computer model.

CHAPTER 4.

TRANSFERRING KNOWLEDGE

Throughout expert system literature, the knowledge acquisition phase is constantly being labelled as the bottleneck for expert system development [7,8,37]. Typically human experts are not used to describing everything they do in connection with solving a particular problem. They may forget portions of information, or unknowingly provide an inaccurate description of their true expertise. Finally, experts may mistrust expert systems or fear the consequences of their development (i.e., loss of jobs) [42]. Eliciting knowledge then becomes a very difficult task, which may represent the largest portion of the development time for an expert systems project. Present methods of acquiring expertise, through human communications, are slow and cumbersome.

However, for any computer model to be used, the results must be trusted. In the case of an expert system, the domain knowledge of an experienced water resources engineer must be properly transferred to the computer, so an accurate representation of his expertise is achieved. Thus, the judged degree of success will depend, in part, on the completeness and correctness of the knowledge base. Certain ingredients are necessary for a successful knowledge transfer attempt.

4.1 Approaches

There are three basic approaches to gathering the information to be contained within the knowledge base of an expert system [29]:

1) train the actual expert in knowledge acquisition techniques and have him develop a conceptual model of his thinking process;
2) train an experienced knowledge engineer in the domain and have him work with the expert;
3) train an engineer, who has previous knowledge in the domain, in knowledge acquisition and have him communicate with the expert.

Usually, the first approach is neither practical, nor advisable for engineering experts. Not only is their time valuable, but it is often very difficult to examine one's own thinking process. An expert may tend to take most of what he knows and does for granted. Training an experienced knowledge engineer in a particular domain may be a costly, and unnecessary operation, especially when the domain is very complex. The drawback of the final approach is the time and effort required to become adequately acquainted with expert systems and knowledge acquisition. However, this engineer will be trained to a degree, both in knowledge acquisition and the problem solving process of the expert, through the development process. This knowledge can then be utilized in future projects.

The recommendation from other engineering expert system studies is that the third approach appears to be the most promising [10,29]. Figure 4 is a schematic of this approach. The popularity of this approach is also due to the existence of easy-to-use expert system tools, many for use on

microcomputers, which simplify the job of developing expert systems.

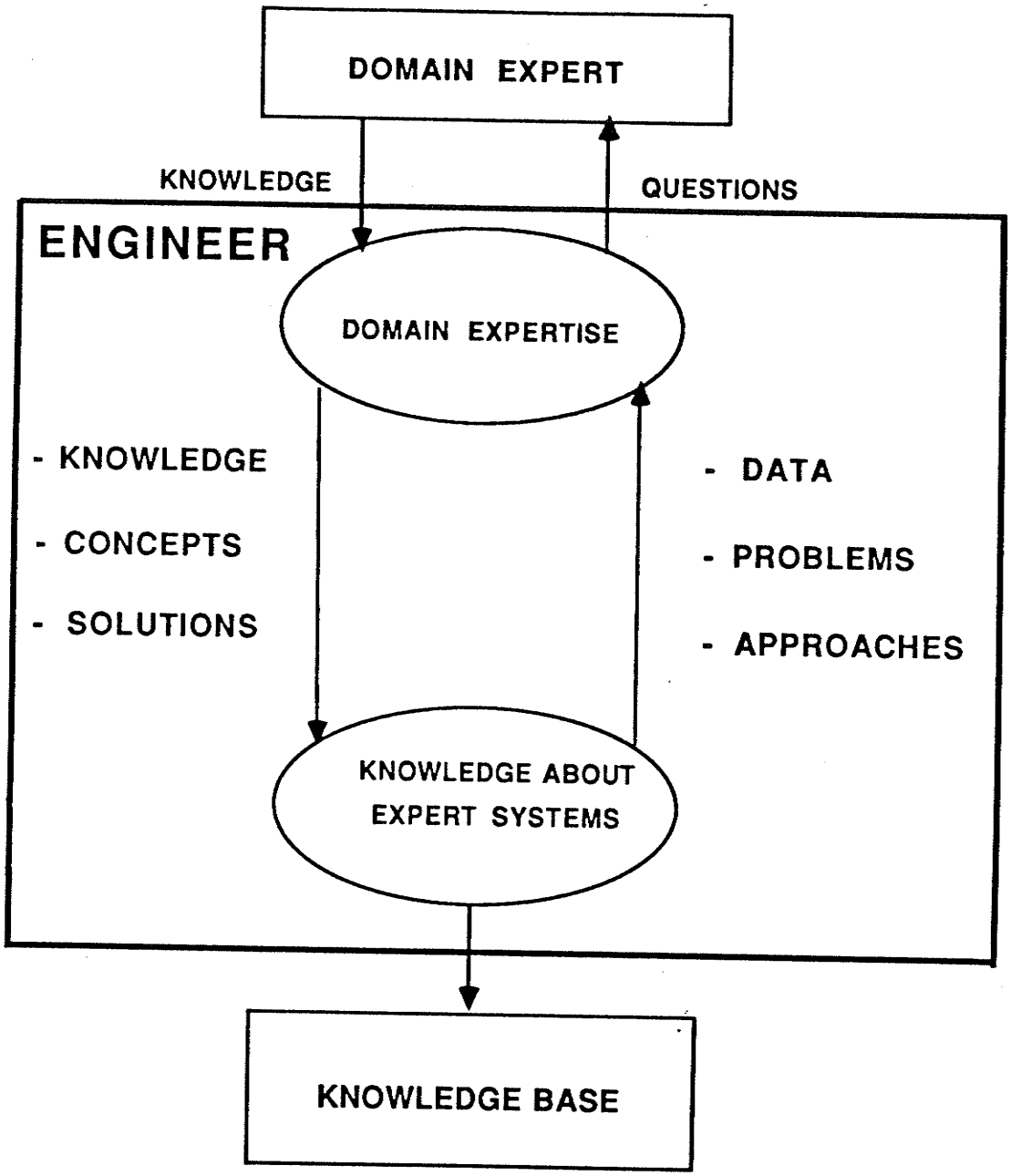

Figure 4. The engineering expert system approach [50].

4.2 Knowledge engineering

It is the duty of the chosen knowledge engineer to form an initial model of the expert's thinking process, incorporating as many relevant facts, concepts, relationships and special cases as possible. This must be accomplished without the luxury of having a complete understanding of the problem domain. The conceptual model is then thoroughly tested, comparing its recommendations and conclusions to that of the expert [25]. An expert system is slowly, and incrementally developed to control the complexity and ensure its correctness, as each new piece of knowledge is acquired and implemented. The knowledge engineering process is, thus, not a simple one-step linear process [18].

Few guidelines are available on issues such as selecting experts or conducting interviews. This does make practical sense, however, as the field is more interpersonal and abstract. A novice knowledge engineer must rely extensively on opinions expressed in literature, by others who have attempted similar projects. Numerous writings on the subject of knowledge acquisition do, though, include a list of techniques for extracting knowledge, assuming an articulate, co-operative, enthusiastic, high-level expert has been identified [19,55]:

1) **direct observation** - watch the expert solve a problem (without interruption), as he "thinks aloud", making notes on the process he uses;

2) **problem discussion** - explore the data, concepts and relationships needed to solve a set of prototypical problems - interrogate the expert and address any

inconsistencies or potential gaps in reasoning that were observed during direct observation;

3) **problem description** - ask the expert to describe typical problems for smaller subsets (categories) of the domain, which aids in conceptualizing the relationships between problems and knowledge strategies needed to solve them;

4) **problem analysis** - use a combination of direct observation and problem discussion on a group of problems - try to uncover potential generalizations and specializations of knowledge and problem solving strategies;

5) **system refinement** - the expert supplies a set of problems to solve using the rules acquired from the interviews - use feedback to establish any missing or inconsistent knowledge, or incomplete problem solving strategies - should be done as soon as possible;

6) **system examination** - the expert examines and critiques the knowledge and reasoning of the prototype system - the resulting information, regarding justifications for knowledge, or reasons why strategies and/or knowledge are not applicable should be documented, so it can be added later;

7) **system validation** - outside experts, end-users or managers evaluate the utility and performance of the system - conduct system validation

throughout development to ensure end-user feelings are considered in development decisions.

A few observations can be made regarding this list of knowledge acquisition techniques. The first four techniques are suitable for the initial uncovering of expertise, while the last three are applied at the point when this knowledge has been represented in the form of rules, or a small prototype system, to be tested and refined. These seven techniques involve a combination of observing the expert solve a problem, questioning him for further clarifications, and relying on him to perform some degree of introspection regarding his thinking process. Both observation and interviewing are inherently slow processes, but can work given a good knowledge engineer.

4.3 Selection of a knowledge engineer

In the development of expert systems, a great deal of thought must go into the selection a knowledge engineer for the project team. As previously mentioned, it is important that he have some previous experience in the domain, but other characteristics are necessary in a good knowledge engineer. Several writings on the topic of knowledge engineering have emphasized the idea that a suitable candidate for this position need not be an experienced computer programmer [25,26]. Rather, this person's job task parallels that of systems analysts, where communication and problem solving skills are more helpful. The idea, after all, is to develop a computer model, in this case a model of the skilled and informed judgement of a human expert solving a difficult problem. If a systems analyst is able to construct a relatively thorough and accurate

conceptual model of a particular engineering problem, the actual programming becomes that much easier. This sentiment applies equally well to the job of knowledge engineers.

Hart, in her book dealing solely with the subject of knowledge acquisition, lists the desirable qualities of a knowledge engineer [26]. These are: good communication skills, intelligence, tact and diplomacy, empathy and patience, persistence, logicality, versatility and inventiveness, self-confidence, domain knowledge, and programming knowledge. No doubt, if such a person exists, he or she would make a perfect knowledge engineer. One must also be aware of the interaction between the knowledge engineer and the expert. A suitable candidate should possess a good number of these aforementioned qualities, and also have a personality compatible with that of the expert. Perhaps, a suggestion is to have more than one person involved in the knowledge acquisition process. It is the combination of expert and knowledge engineer that will ultimately decide the extent to which his knowledge can be transferred to a computer-based form. This statement applies outside the realm of expert system development, as well.

4.4 Standard training process

In the knowledge acquisition arena, new terms have been applied to established concepts. Daily, people perform 'knowledge acquisition' and the duties of a 'knowledge engineer', without being aware of doing so. Whenever anyone begins a new job, for example, some level of training is provided. Usually, the first step is locating someone who knows what they are doing, i.e. an expert, who is willing to spend time teaching new

employees. The trainee will then watch the expert perform the task, and ask for clarifications or explanations.

There are no strict guidelines on performing a successful transfer of knowledge, whether it be for on-the-job training, or expert system development. The success depends, to a great extent, on the ability of the knowledgeable person to articulate and explain the necessary information at the right level of understanding, and the ability of the inexperienced person to understand and learn within the domain and to ask the right questions at the right time. Not everyone makes a good trainer, and not everyone is trainable. It is, therefore, often difficult to pinpoint the underlying reason for a failed attempt at knowledge transfer.

Expert systems can be viewed as, simply, an additional step in the standard training procedure. The basic idea is to train the knowledge engineer in the domain, and for him to transfer, in turn, what he has learned into some computer usable format. As such, the knowledge acquisition phase is prone to the same difficulties as any commonplace expert-apprentice scenario. The knowledge engineer may be confronted with the paradox of expertise, whereby the more competent an expert becomes, the less able he is to describe the knowledge and reasoning used to solve a problem [55].

It is true, however, that the transfer of knowledge is not simply an ad hoc procedure. Often times, through experience and increased problem understanding, companies have been able to establish standard training programs that have been found to work quite well. The difficulty with establishing guidelines for expert system development comes from the diversity of domains, problems, and knowledge sources involved. For

expert system development, one tries to speed up the training process, making it more essential to have good people on the project team. These project members can likely benefit, though, from hearing or reading about the experiences of other expert system development teams.

In the end, the procedures are all commonsense, however, due to the human factor, the process is anything but straightforward and simple. Bramer [8] points to the lack of emphasis on techniques, or problems of knowledge acquisition in "popular" literature, making it possible for the "unwary to conclude it possesses no difficulties" [8]. Through this thesis, and the sharing of project experiences, it is hoped that others can benefit from some of the suggestions and personal opinions expressed. The following sections will introduce and describe the Manitoba Hydro project. This project will then be used to describe the steps involved in knowledge acquisition.

CHAPTER 5.

MANITOBA HYDRO CASE STUDY

5.1 Problem related objectives

The main research objective was to investigate the combination of knowledge acquisition for expert system development and water resources engineering expertise. A case study was undertaken to create expert systems that could act as "intelligent" interfaces to a computer model used at Manitoba Hydro. Manitoba Hydro looked at not only the benefits to research, but possible benefits to their company. They view expert systems as a means of capturing extremely scarce knowledge possessed by one of their engineers, to both assist him and ease the burden of teaching or training others to develop expertise in this particular domain.

5.2 The Reservoir and Energy Scheduling Engineer

The computer model that is the focus of this research is known as EMMA (Energy Management and Maintenance Analysis). The model is a decision making tool used by the Reservoir and Energy Scheduling (RES) Engineer, at Manitoba Hydro. It is the responsibility of this engineer to prepare an operating schedule for Manitoba Hydro's 13 reservoirs, 13 hydro generating stations, 2 thermal generation plants, and 9 inter-utility connections. Much of the information regarding the Manitoba Hydro system and EMMA was derived from talks with the RES Engineer, two conference articles [4,12], the EMMA user's manual [13], and work by Reznicek [47].

Manitoba Hydro has a number of objectives it wishes to meet

through the operation of its integrated power system. The following is a list of objectives, in order of priority, supplied by the RES Engineer that must be considered in his decision-making:

1) winter energy supply must exceed the forecasted Manitoba load even under the most severe winter weather conditions;
2) ensure sufficient summer energy supply;
   (Deficiencies in energy supplies can be purchased through the import capabilities of the system, if need be.)
3) ensure reserve energy is available within the hydraulic system to satisfy any firm energy commitments in subsequent years, under the worst drought conditions;
4) maintain winter operating reliability, so as to avoid a system blackout during winter months;
5) maintain summer operating reliability, to avoid a summer month blackout;
6) meet social responsibilities to other lake or reservoir users - environmental requirements of fish and other wildlife, concerns of other businesses and recreationalists regarding lake levels or fluctuations in levels, adhere to agreements with outside control boards - the Manitoba Hydro system is operated in such a way as to minimize the adverse impacts on communities;
7) operate as economically and efficiently as possible.

He must also contend with the various constraints acting on the hydraulic and electric systems, including:

a) physical constraints - water supply, fuel supply, system capability, reservoir sizes, rating curves;

b) regulatory constraints - licenses, agreements, environmental orders restricting what actions can be taken at specific sites;

c) practical constraints - how close to the hydraulic and electric system limitations can you operate;

d) management constraints - budget, risk.

5.3 The EMMA computer model

Due to the complexities of the Manitoba Hydro system, a computer model was developed to assist the RES Engineer. EMMA (Energy Management and Maintenance Analysis) is a deterministic optimization model that can model hydraulic, electrical and maintenance systems. EMMA can be used for short-term operating decision making or more long-term planning. It, thus, incorporates variable time horizons and time steps. EMMA, itself, calculates a schedule of reservoir releases, a power generation schedule, and an import/export schedule. This tool is said to give the RES Engineer flexibility in describing his operating objectives and constraints. It then figures out how to operate the system in order to achieve these specifications. A more detailed explanation of EMMA is provided, as subsequent sections involve descriptions of the knowledge acquisition procedure and prototype systems for this project. These descriptions include references to specific aspects of EMMA.

The hydraulic system is composed of reservoirs, lakes and rivers. Reservoirs generally have two possible outlets: a spillway; and a penstock,

to route it through a power plant. Lakes, on the other hand, may be controlled by natural outflows or through a control structure. The storage capacity of lakes or reservoirs can be divided into segments to approximate nonlinear stage-storage curves, and the final segment can also be divided up into figments. The following constraints can be formulated for the hydraulic system, within EMMA:

1) **sum of figments** - to ensure that the sum of the figment storages is equal to the storage in the last segment;

2) **flow continuity** - to ensure that, for any lake or reservoir, the change in storage is equal to the difference between inflows (upstream penstock discharges, upstream gated control or spillway flows, and natural inflows) and outflows (penstock discharges, gated or spillway flows and natural outlet flows);

3) **run-of-river** - links the operation of a hydro generating station with an upstream generating station for strips of a load duration curve and does not permit a change in storage to take place;

4) **same stage** - to set the initial elevation of a lake to be the same as the final elevation;

5) **generating station flow bound** - to limit the outflow from a spillway or powerhouse as a function of upstream storage;

6) **control structure flow bound** - to limit the outflow from a control structure as a function of upstream storage.

The electrical system consists of the components required for energy

generation and transmission to meet the specified load. The domestic load is represented by a load duration curve, and is divided into strips, so that an average load is assumed, and must be met, for a specified duration. Strips are designated as on or off peak, as energy prices vary according to these classifications. Energy can be generated to meet demands through hydro and/or thermal generating plants. Energy can also be transferred to or from neighbouring utilities. These import and export commitments may be either firm (must be satisfied) or interruptible. The constraints that can be formulated for the electrical system are:

1) **energy supply and demand** - to ensure that for every load duration curve strip, the system load plus firm exports are supplied from hydro generating stations, thermal generation sources and imports;

2) **hydro production** - to convert the powerhouse discharge over a timestep into energy for each load duration curve strip;

3) **hydro shaping** - to ensure the capacity utilized at a hydro generating station, in the load duration curve: decreases moving from on peak to off peak; or remains the same if the station is base loaded (i.e., forced uniform energy generation);

4) **generation source shaping** - the same as 3) except it applies to thermal plants;

5) **peaking** - to limit the ability of hydro generating stations to peak by the maximum change in storage or drawdown permitted across on-peak strips of a timestep;

6) **tieline total load** - to ensure that the sum of imports and exports (firm and interruptible) does not exceed the capacity of the transmission line;

7) **import fraction limit** - to limit the total interruptible (not firm) import energy to a percentage of the required system energy for each timestep;

8) **tieline minimum energy** - for firm energy contracts, a minimum energy must be delivered;

9) **grouping constraints** - to place a minimum (as a function of the system load) on the total of the defined energy for a desired group of energy sources (hydro generating stations, thermal plants, exports, imports) over strips, timesteps, or period of study;

10) **waste heat** - to limit the maximum capacity of a thermal plant when thermal pollution of downstream water exceeds a specified maximum temperature or maximum temperature change.

Finally, EMMA models maintenance systems to: allow an annual maintenance schedule, based on station maintenance requirements and system operation requirements; and review, on an ongoing basis, a predetermined maintenance schedule. Data provided regarding maintenance scheduling includes the preferences of station staff on maintenance periods. EMMA offers the formulation of the following constraints with regards to maintenance scheduling:

1) **crew scheduling** - to ensure that, for a specified crew, holidays, etc., are accounted for;

2) **required maintenance** - to ensure that for each plant (hydro or thermal) sufficient maintenance is done to meet the set requirements;

3) **maintenance space** - to ensure the peak load can be met after maintenance is accounted for and an allowance is made for forced outages;

4) **available capacity** - to ensure that for each plant (hydro or thermal) the available capacity is not exceeded by the energy generated plus scheduled maintenance;

5) **crew availability** - to ensure that maintenance done by a crew within a timestep does not exceed their availability in hours.

The objective function of EMMA contains the variable values associated with energy, storage, releases, and maintenance, and respective cost coefficients. The cost coefficients are used to reflect a number of issues [13]. The storage cost coefficients may reflect:

- flood damage on upper lake or reservoir segments;
- the value of water in storage to future energy generation;
- a penalty cost to ensure that EMMA fills the bottom segments of a lake first.

The value of the cost coefficient assigned to releases may reflect the net benefit or cost associated with a flow assignment. Cost coefficients related to generated energy reflect fuel costs and the fixed costs of maintaining the plants. Export and import energy cost coefficients may reflect the structure of the interruptible market and the firm contract

energy prices. Finally, the cost coefficient associated with scheduled maintenance may reflect non-economic scheduling preferences.

Along with the comprehensiveness and flexibility of the EMMA modelling tool comes complexity. The RES Engineer must possess several types of expertise. He has collected and compiled problem-related theoretical knowledge of the technical aspects of the physical systems. As well, EMMA is based on linear programming, and, therefore, approximations, linearizations and segmentation of data, such as stage-storage and load duration curves, is necessary. The RES Engineer must provide EMMA with data consistent with the physical and time structures of the problem he is running. He relies on a deep knowledge of mathematical programming and modelling within EMMA, not only in formulating a problem, but in analyzing the outputs.

There are several issues the RES Engineer must contend with at the output analysis stage. First, as a linear programming tool, infeasibilities may arise when running an operation or planning problem in EMMA. The cause must be identified and dealt with, if possible. Even when a feasible solution is reached, the RES Engineer must review the overall viability and practicality of the solution (i.e., can you realistically hope to achieve the releases and power generations called for?). Finally, he must examine the environmental, social, and economic impacts of such a plan. Some of these issues may be integrated into the problem formulation, others rely extensively on judgment and experience. The RES Engineer has developed a feel for the physical meaning and "rightness" of many values. In view of these characteristics, the logical choice was to develop expert systems around the EMMA model to act as "intelligent" interfaces.

The EMMA computer model allows the RES Engineer at Manitoba Hydro to incorporate more information into his decision making. In order to do this, the model had to be very comprehensive and flexible. To date, only one person at Manitoba Hydro, the RES Engineer, has the necessary experience to properly formulate and run their planning problem using the EMMA model. Thus, a joint project between Manitoba Hydro and the University of Manitoba is underway that is looking into the possible integration of expert systems into the RES Engineer's problem solving environment. The idea is to create "intelligent" pre- and post-processors to the EMMA model to assist the expert and also act as training, or learning, tools for less experienced EMMA users.

5.4 Research objectives

Through this research, it was hoped that more could be learned about the type of knowledge an experienced water resources manager might possess. The characteristics and structure of the expert's knowledge may have consequences on the capacity for transferring this information to an expert system.

Thus, an attempt was made to apply the techniques and approaches gathered through an extensive literature review to a "real-life" problem and expert. The following section describes the procedure undertaken, some problems encountered, and how, through hindsight, some of these troubles might have been avoided. These findings provided, as well, a partial footing from which to construct suggestions regarding expert system development and knowledge engineering, in the more general application area of water resources engineering.

CHAPTER 6.

KNOWLEDGE ACQUISITION PROCEDURE

6.1 Learning about the domain

Before anything else can be done, the knowledge engineers have to gain a basic understanding of the problem domain. Prior to this extensive research should be conducted into the area of expert systems. Thus, initial effort was expended, by the knowledge engineers (myself and a fellow civil engineering graduate student) in "getting to know EMMA". It was felt that the burden of understanding such a complicated model as EMMA and the extremely difficult tasks performed by the RES Engineer could be reduced as both knowledge engineers were also trained civil engineers. The knowledge engineers were aware of the fundamentals involved in reservoir management and linear programming, though they had no work experience in these areas.

Usually, it is easiest to begin this learning period by examining the available static knowledge on the subject, before attempting any direct discussions with an expert. Static knowledge refers to such things as textbooks, manuals, or case study reports. For this project, the EMMA manual and articles written on the model were studied. Knowledge acquisition literature emphasizes the importance of this initial learning period, as the knowledge engineers gain a grasp of the terminology and problem concepts [9,31]. One cannot rely on the expert's initiative to spell out rules and determine what must be known. Experience was also gained by examining an existing input file and running the EMMA model. The objectives, constraints and variables could then be studied in greater

detail.

This learning process should not be restricted to the expert system developers alone. The expert should be educated about expert systems and what is involved in developing them. The knowledge engineers conducted several informational seminars open to any interested university or Manitoba Hydro personnel. Still, even after an extensive review of available literature, the expert and knowledge engineers all had a quite naive and overly optimistic view of expert systems and what could be accomplished in the first year of development. This blind optimism may be due partially to the lack of "bad experiences" reported in the literature. Often times, only success stories are reported. Bramer, in his article on expert systems [8], noted that it may be "all too easy for newcomers ... to misjudge it (expert systems) and make claims that are overblown or even patently absurd".

A very important point must be mentioned regarding the attempts to understand all the components of the EMMA modelling tool. The approach taken by the knowledge engineers, to EMMA, did not allow them to gain a physical understanding of the Manitoba Hydro power problem. Again turning to the available literature on knowledge acquisition, it is often stated that knowledge engineers need not have a background in the domain they are investigating [14,18,54]. For most engineering applications, this statement will not really be valid. After all, it is hard to model the unknown. The complexity of the Manitoba Hydro domain and the time constraints on the project prohibited the knowledge engineers from gaining a full understanding of the EMMA planning problem. A lack of problem understanding hampers future interactions

with the expert.

6.2 Constraining the problem

An expert system project rests on one standard and accepted principle, that being **incremental development.** A project begins with the development of a prototype system that will point the way for future work [6,14]. This system will typically be very small, and, truthfully, not very expert. The idea being, that the knowledge base is slowly and carefully expanded to deal with more complexity and involve more expertise. Since incremental development is the key, a prototype should be built as soon as possible to obtain the necessary feedback for further expansion. Expert system development tools facilitate rapid prototyping, and, thus, the feedback approach.

The incremental approach requires the selection of an appropriate scope for the initial development of a prototype system. The scope should reflect the true nature of the expertise and the problem solving approach required within the domain, to provide any insights into future development. Usually, and especially with engineering applications, a problem will be too large to look at in its entirety. It is necessary, then, to constrain the scope by either looking at relatively small, simple, but whole problems, or looking to a greater depth at specific subproblems or subtasks.

Unfortunately, the knowledge engineers, at the beginning of the project, have limited personal domain experience on which to base the choice of a scope. Thus, reliance is placed on the expert's own judgement and examples found in the literature of similar projects. The

job of the RES Engineer is much too difficult and involved to assume the creation of one expert system can encompass all he knows. Designing expert systems around an existing computer model simplifies the task of selecting a scope for the prototype, as there are two major divisions- data preparation (parameter estimation, data analysis, model options selection, etc.) and output analysis.

The knowledge engineers quickly became aware of the enormous amount of private knowledge the expert possesses with regards to both the integrated Manitoba Hydro power system and the modelling tool EMMA. Through initial discussions with the expert, an overview was established of the various tasks he performs. With his help, two seemingly appropriate subtask areas were identified as potential applications for prototype expert systems. The expert conducts checks on important input and output variables and is able to quickly recognize any unexpected or inappropriate data. This is particularly important on the output side, when he must judge the practicality of the operating plan EMMA prepares. The second area, one in which the expert seemed quite interested, was the handling of infeasibilities. It takes a great deal of experience to know how to deal with running an apparently properly formulated problem and having no feasible solution possible. The culprit may be a simple input data entry error, or a poorly formulated problem, or not having enough resources available to satisfy the power demand. A user would like to efficiently and effectively search for the underlying cause of an infeasibility.

6.3 Knowledge acquisition sessions

After developing the necessary background in the problem domain and restricting the initial development scope, the knowledge engineer begins more direct and intense interactions with the expert. Through discussions, he attempts to set out the manner and resources with which the expert works in his search for an acceptable solution. It was decided that discussions would be documented by taking notes. Halfway through the knowledge acquisition process, a regular meeting time was established. It is a wise idea to establish a regular meeting time with the expert from the start, so he is able to plan around the sessions, even if some of the meetings are less formal, and simply more of coffee break chats.

During the development of the prototype expert systems, it was necessary to rely on a number of approaches to gathering information about the problem. For the infeasibility subdomain, the expert suggested the best approach may be for the knowledge engineers to run the model themselves, using an existing input file and changing individual input data items trying to create infeasibilities. EMMA, in one of its output reports, lists the variables and constraints involved in the infeasibility. The goal was to establish a link between the input data modified and the variables and constraints indicated by EMMA. Because of the number of components in the power system, the input file was large, and runtimes could be very long. To reduce the problem, and also for the benefit of understanding, it was decided that a subsystem of the entire Manitoba Hydro system would be considered. Unfortunately, the expert did not have much experience with this artificial planning problem. A better approach may have been to

stick with the larger, more realistic problem, so better use could have been made of his expertise. However, regardless of this factor, an obvious pattern between the specific input data change made and the output variables or constraints indicated was not observable.

Consideration was given to using the rule induction capacity of the expert system development tools under investigation. However, due to the large number of parameters (possible constraints and variables that could be indicated) this capability was not explored. As well, depending on the actual value set for a particular input value (i.e., a large versus a very large value), the resulting set of indicated constraints and variables could change. Certainly, if a good set of example cases is available, then machine induction can be a very useful tool for rule generation [8]. With regards to the intelligent data checks performed by the expert, knowledge was elicited through the standard question-and-answer approach. One is bound to run into a few obstacles during the knowledge acquisition phase. The next section describes some of those that were encountered during this project.

6.4 Lessons learned in knowledge acquisition

One of the major obstacles, to obtaining a clear picture of what the Manitoba Hydro expert does, was simply the lack of a proper problem understanding by the knowledge engineers. At first, all the communication difficulties with the expert were attributed to this factor. However, more recent efforts by the expert and another experienced Hydro engineer, revealed a further error in judgement that likely contributed to the troubles. The initial prototype development scope

appears to have been poorly chosen. The two Manitoba Hydro engineers set out a more precise conceptual model of what the RES Engineer is trying to accomplish with the help of EMMA. They identified a series of decisions, made prior to the actual data checking and output analysis, that can have significant impacts on these activities. Thus, this first year has been a learning experience not for the knowledge engineers, but also for the Manitoba Hydro personnel involved. This is one of the benefits of an expert system project, as it may ultimately provide the expert with more insight into the problem he faces and his solution procedure.

The identification of this hierarchical model, represented in Figure 5, of the RES Engineer's job was a major breakthrough in the project, that occurred too late to influence the first year of development. The first level of decision-making with EMMA, or any computer model for that matter, is the motivation for using the model in the first place. Are you considering short-term planning, long-term planning, or budget preparation? This decision will impact the objectives of the run, and, thus, the input data required, for example expected versus worst-case inflow forecasts, and the subsequent format of the input file, for example the planning horizon or timesteps considered. These upper level decisions will have a significant effect on which input and output variables are important in the analysis, and the possible infeasibilities that might arise. For the initial prototype development, the scope was restricted to the bottom level of this hierarchy, and, thus, connections with prior decisions were unknowingly severed.

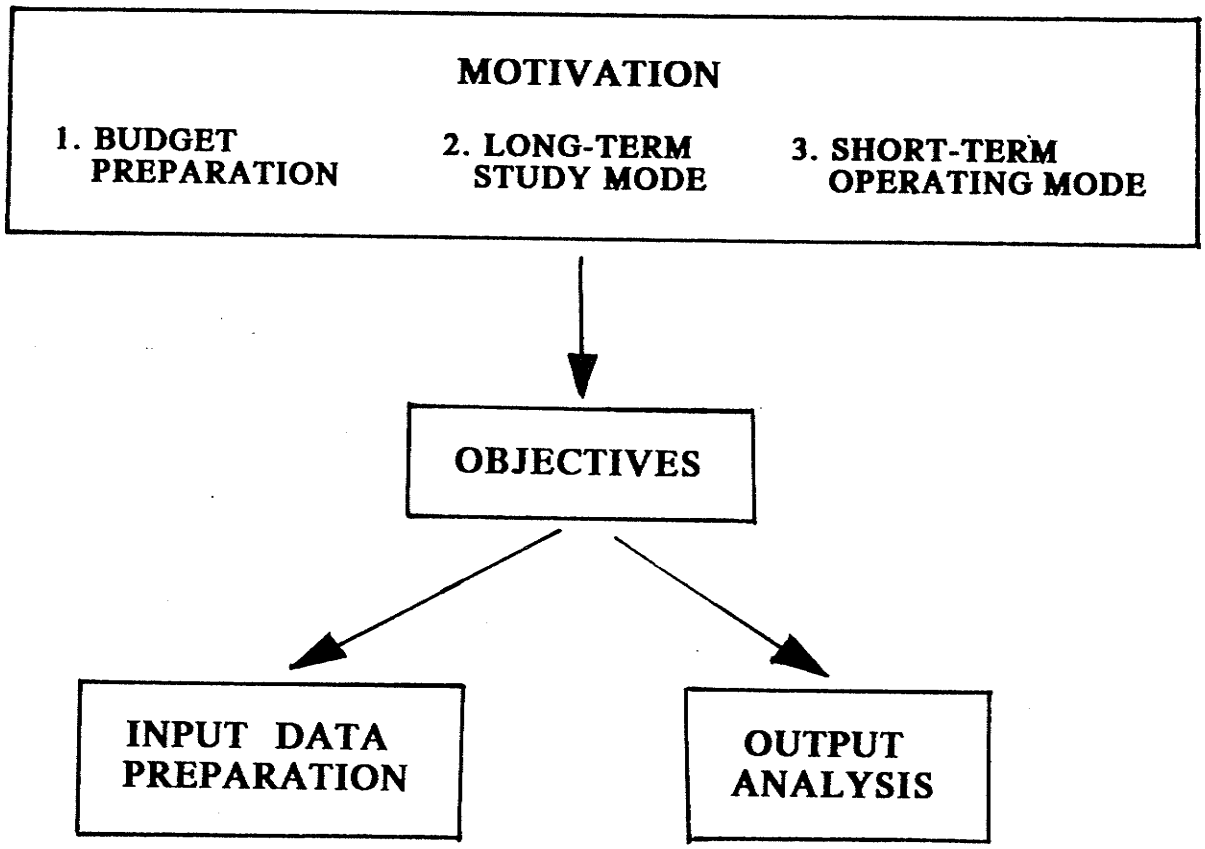

Figure 5. Hierarchical model of problem formulation in EMMA.

It is easy, through hindsight, to recognize the importance of defining a conceptual model of the problem solving right at the start. Much effort and time can be saved, if enough effort and time is devoted to the problem of defining the initial development scope [19]. This step must not be treated lightly, as it can often determine the ultimate success of an expert systems project.

The immediate effects of the lack of domain understanding by the knowledge engineers and the poorly chosen scope were most obvious when the actual knowledge acquisition sessions with the expert were begun. Both the expert and the knowledge engineers could not help but be a little discouraged by the lack of significant progress. The knowledge acquisition literature stresses the need for very structured and focused sessions, and rightly so [19]. Any expert will have very compiled knowledge and will need a fairly narrow context on which to concentrate his attention. Otherwise, as with some of the Manitoba Hydro project knowledge acquisition sessions, the questions being asked can be too general. The expert, in response, gives very broad and general information. In this manner, true expertise is rarely uncovered and any subsequent prototype system will be of little practical value.

Another suggestion for alleviating the problem of less-focused and structured interviews is for the knowledge engineers to make an attempt at interpreting what was learned during a particular session. Consequently, the initial prototype systems reflect more the problem understanding of the knowledge engineer than that of the expert. It is here that the principle of incremental development should be applied. The expert, when presented with some rules or a small system, will be

able, through criticism, to express his true problem solving expertise. This project would have benefitted tremendously if the expert had been kept more up-to-date on the development and been given regular access to the systems developed. It is often too easy to refrain from doing so, especially, as experts are usually extremely busy. This is where the persistence of the knowledge engineer comes into play. He must allow the expert to examine his interpretation of the expert's problem solving approach. In other words, testing is a critical step in the knowledge acquisition process.

6.5 Testing

Testing cannot be strictly separated as a phase in the development procedure, rather, it occurs continuously throughout development. As each new piece of knowledge is learned from the expert, it is added to the system and the expert must then perform a detailed evaluation of the system. Testing becomes a more rigorous endeavour, as the performance level of the system increases. Expert system tools are designed to complement this incremental development approach. Through testing, any errors or bias should be removed from the system. At later stages, this process should include the viewpoints of outside experts and end users. During testing one should also attempt to observe the limitations of the system. Don't just test what you know the system can handle [19]. The expert can be queried as to why a certain problem is not suitable for solution by the expert system. Such information may be helpful in advising a user about the suitability of his particular problem to the expert system.

6.6 Importance of the prototype to development

Due to the nature of expert systems, providing advice by relying on its knowledge sources, it is essential that prototype systems provide extensive help to a user. An even more crucial component of expert systems, what really sets them apart from more conventional programs, is their ability to explain their actions. An end-user will not likely blindly follow, or even accept, the expert opinion provided by an expert system if it is not accompanied by some explanation as to how this conclusion was reached. As well, much of the knowledge required by a user will be implicit in the rules, and only become available through the explanation facility that often backtracks through the rules used to arrive at a conclusion. Thus, major focuses in the development of these computer systems are the help and explanation facilities. The intent should be not only to model the expert's problem solving procedure, but also offer the help and explanations that might be expected from any human expert. Part of the knowledge acquisition process is to identify appropriate help messages and ensure the reasoning steps taken by the system truly reflect the behaviour of the expert.

The incremental development approach has another more practical benefit. It permits a constant evaluation of the progress of the project, and allows for an informed judgement to be made regarding future development. It was only by developing the prototype expert systems that the Manitoba Hydro project team were able to identify an apparently more suitable and more achievable project focus. The knowledge that was learned in this first year is not lost, however. It will simply be incorporated, as needed, into the recently developed hierarchical

conceptual model of the reservoir and energy management and planning problem at Manitoba Hydro. Developing a prototype system can make it painfully clear if a problem is ill-served by expert systems, or a project scope has been poorly chosen. It also helps direct development along a more suitable path, as both expert and knowledge engineer become better acquainted with expert systems.

This increased awareness of expert systems by the project participants often only occurs after first-hand experience. Previous expert systems projects, reported in the literature, do serve as an information source. However, in the expert system field, there are no real guidelines for choosing projects, tools, or development teams that will ensure success. Anyone new to the field will have to, almost inevitably, learn by doing. Part of this learning process is obtaining a more realistic view of the capabilities of expert systems.

6.7 Limitations of expert systems

Even after thorough testing, an expert system will still have certain inherent limitations on its capabilities [7,55]. Human experts have the ability to reason in time and space, which is difficult to simulate in a computer system. The area of an expert system's expertise is very narrow and they exhibit rather fragile behaviour at the boundaries of this knowledge area. After all, an expert system lacks commonsense, or general, knowledge, and also lacks creativity. Such a system cannot, therefore, adapt to changing circumstances beyond the scope of its knowledge. The exhibited power of an expert system is derived solely from the information contained within its knowledge base.

An appropriate view of expert systems, in light of these limitations, was expressed in an article by Blockley [5]. He refers to the term expert systems as a misnomer, and proposes the term "advice systems" in its place. An expert system can never be truly expert, as it is not a responsible decision maker. These systems can only advise. It is up to the user to judge the appropriateness of its advice, while considering any other available sources of information. For this reason, these "advice systems" must be equipped with elaborate help and explanation facilities. It would be foolish to accept the advice given by any source without any justifications, and base an engineering decision on this advice. Thus, expert systems cannot replace engineers, as creative and responsible decision-makers. They simply provide access to the opinion of one expert (or perhaps several) to allow an inexperienced user the luxury of making a more informed, and, therefore, more intelligent decision.

CHAPTER 7.

PROTOTYPE SYSTEMS

The difficulty and associated results of an expert systems project can vary greatly. To contrast the complex and large-scale Manitoba Hydro reservoir management application, the development of a prototype expert system in the preliminary dam design domain will be presented. These two projects differed in several very important ways. Firstly, and most critically, the Manitoba Hydro Project involved the acquisition of knowledge from a human expert, while the knowledge base for the dam prototype system was derived from a written report, produced by the National Research Council (NRC). This text basically represented the results from a previous knowledge acquisition attempt. Design experts in the United States were interviewed, and the resulting information was organized and compiled into report form.

Due to the difference in the knowledge acquisition procedure undertaken, in connection with the two projects, the project teams were different. There was only one person involved in the development of the preliminary dam design prototype. On the other hand, two knowledge engineers participated in the Manitoba Hydro project, and a full time computer expert would have been a great benefit to the project team.

Additionally, the Manitoba Hydro project knowledge engineers had to develop an understanding of the reservoir management and Manitoba Hydro domains, in order to communicate with the expert. It was impossible, in this case, to avoid immersing oneself in the complexities of the application domain, as could be done for the preliminary dam design

case study. The preliminary dam design example simply involved transferring knowledge from a text to an expert system. However, little documentation was available describing, in enough detail, the tasks undertaken by the RES Engineer at Manitoba Hydro. The only possibility, therefore, was for the developers to gain a deeper understanding of the domain than was necessary for the other project. Learning within a difficult domain such as reservoir and energy management is a slow process.

7.1 Manitoba Hydro prototype systems

The effort extended in the first year of the Manitoba Hydro project has produced two prototype knowledge-based systems. The systems were designed to act as "intelligent" interfaces to the EMMA computer model. A pre-processor and a post-processor were both developed within version 2.0 of RuleMaster, an expert system tool compatible with the Apollo computer, on which EMMA is running. Versions of these systems were subsequently transferred to two PC-based expert system development tools, Personal Consultant+ and VP-Expert. All three development tools represented knowledge in the form of rules. It was possible, only with the RuleMaster modules, to integrate the prototypes with the EMMA program, through the C programming language. The following sections describe the contents of the pre- and post-processors. A complete listing of the pre-processor is included in Appendix A and the post-processor listing can be found in Appendix B. These listings include any necessary C functions. The knowledge found in these programs will be explained in the following sections. When appropriate, references will be made to the

portions of the listings where this knowledge is stored.

Both systems were created to deal with the Laurie River system, a subsystem of the entire Manitoba Hydro integrated power system. It involves: 5 lakes, 2 hydro generating stations, a thermal plant, 4 local inflows, 2 control structures, 1 natural outlet, 1 tieline, and 5 timesteps (months). For the most part, when historical or advisable values or ranges for certain variables in the system were needed for the prototypes, made-up values were used. The point was simply to show that the prototype systems were working properly, the actual values were not important.

7.1.1 Description of the pre-processor

The prototype pre-processor, described in Figure 6, performs a form of "intelligent editing" on a given input file, by comparing some important input data with their historically observed counterparts. The pre-processor has been connected to EMMA, and, thus, accesses the required input values through the data structures created by EMMA itself. Values set for natural inflows, starting reservoir levels and energy production coefficients are checked against observed historical values. Any potential problems, (i.e., values falling outside of historical ranges), are flagged. The user can then ensure the correctness of these values given the fact that they have not been, or have rarely been, encountered in the past. An expert EMMA user will have a feel for these numbers and perform such an analysis without having to, necessarily, refer back to historical data files.

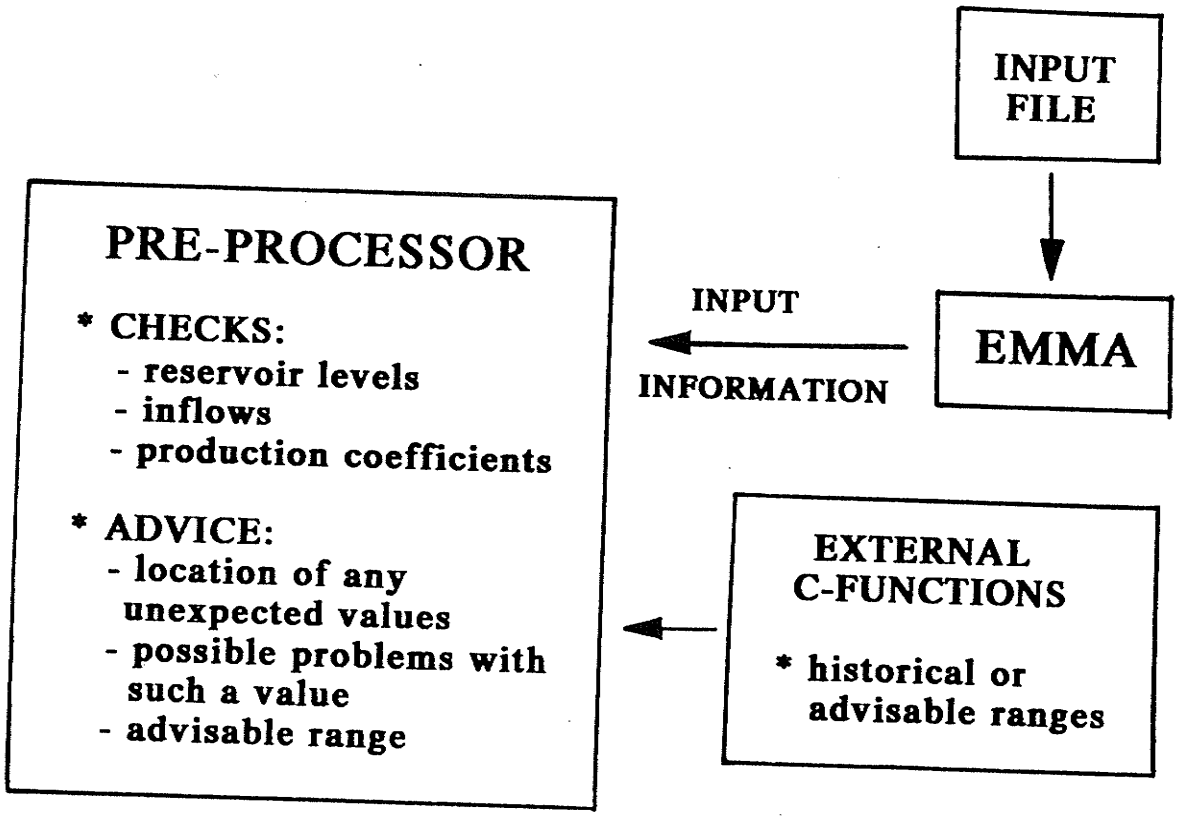

Figure 6. The pre-processor to EMMA.

Prior to this checking process, the user is questioned about the general type of problem he is considering and given initial advice regarding objectives, droughts and worst case scenarios. The user is advised on the use of cost coefficients as a method of accomplishing a special objective. The example provided by the expert was the minimization of spill at a particular spillway, perhaps due to the present

condition of the structure. On the same lines, if the user enters that the system is in a drought condition, water may be kept in storage by adjusting the cost coefficients, so EMMA tries every other means of meeting demands before using the hydro generating capacity. Finally, from discussions with the expert, it was evident that analyzing a worst case scenario was important. Therefore, the user is given the suggestions supplied by the expert regarding the creation of a worst case scenario, through the assumed inflows and system loads. File **input.ind** of the pre-processor is responsible for this preliminary advice.

This initial portion of the pre-processor is really in its infancy stage. The expert did not have a chance to properly examine these initial statements, although the knowledge was derived from the notes taken in the knowledge acquisition session. The inclusion of this knowledge becomes more appropriate when considering the current hierarchical model of the expert's use of EMMA. Objectives are related to the specific problem context one is working with (i.e., planning, operation, or budget preparation).

The input values selected for the "intelligent editing" process were those considered extremely important by the expert. It was also from the expert that the knowledge engineers learned of the type of historical data used for comparisons. Inflows can be compared to the 95 and 99 percentile values of the historical inflows, assuming these flows to be lognormally distributed. The user is simply made aware of the location, in the input file, of any inflows that fall outside either of these ranges. File **flow.ind** of the pre-processor checks the inflow values, using historically derived means and standard deviations. For the prototype system, the

knowledge engineers simply made up values for these parameters for the 4 inflows in the Laurie River system, which were stored and accessed through C functions.

The energy production coefficients specified for each hydro generating station are compared with an advisable range. Since EMMA is a linear program, the formula for energy from a hydro generating plant, which follows, cannot be modelled directly [13]:

$$E = QhK \tag{1}$$

where: Q = discharge
h = head
K = unit conversion factor

The value of head is a function of the discharge, Q, and both head and discharge are decision variables. To pose a linear programming problem, the product of head, h, and the conversion factor, K, is assumed and then checked for accuracy. This product is referred to as the production coefficient of the power plant. From the expert it was learned that: estimates are usually based on the normal operating head of the plant; reasonable limits for these coefficients can be obtained from the range of the storage curve for the reservoir; and starting estimates can have a significant impact on the convergence of the iteration procedure.

As a result, the pre-processor prototype system checks the values of the production coefficients for hydro generating stations included in the problem description (i.e., input file). The user is supplied with the location of any 'bad' estimates and given the reasonable range for the coefficients, and warned of the possible consequences on the iteration procedure. The reasonable maximum and minimum values were calculated

according to the expert's advice, using the limits of the storage curve of the associated reservoirs. These values, for the Laurie River system stations, were then stored and accessed through C functions. File **pcoeff.ind** of the pre-processor is responsible for checking the production coefficients.

Similarly, starting reservoir levels are compared to the minimum and maximum recorded levels for any particular reservoir. These limits are not necessarily equal to the physical size limitations of a reservoir, and a value outside this range would not be expected. The question to be asked is why the user is starting a reservoir at some extreme value. Perhaps, it was an unintentional error. Thus, the user is supplied with the location of the unexpected value and the historically observed limits. Again, to test the prototype system, historical values were made up for the 5 lakes in the Laurie River system and stored in C functions. The file responsible for these checks is **stages.ind** of the pre-processor listing (Appendix A).

Thus, for all of the input data checks, the user is provided with: the location of the unexpected value; the advisable range; possible problems this value may cause; and, in some cases, suggestions on a more appropriate value. The pre-processor prototype knowledge-based system presently contains approximately 50 rules, some of which are simply procedural, and direct the reasoning process. The necessary historical ranges, or values, are stored in C functions which have been connected to the modules developed in RuleMaster. The following is an example of a rule in the pre-processor. It is used in connection with a check of the production coefficients in the input file. The presentation of

this rule is followed by a description of the post-processor to EMMA.

EXAMPLE RULE:

```
IF (production_coeff_min > input_production_coeff) IS

"T": (s_advise "The value set falls below advisable.
                May cause convergence problems.";

        print generating_station,
              timestep,
              load_curve_strip,
              realistic_minimum,
              input_value)
```

7.1.2 Description of the post-processor

On the output side, a prototype knowledge-based system has been created to advise a user of certain issues related to infeasible and feasible solutions. Figure 7 is a schematic of the post-processor and its connection with EMMA.

Assuming EMMA was able to reach a feasible solution to a problem, the system helps the user assess the practicality, or viability, of the resultant operating plan. The RES Engineer can rely on his extensive experience to judge the practical attainability of certain values called for in the solution.

Initially, however, the prototype expert system helps the user perform a few quick feasibility checks. The number of production coefficient iterations is checked against the maximum specified for the problem. If they are equal, the problem may not have converged yet. This rule was developed personal experience running EMMA.

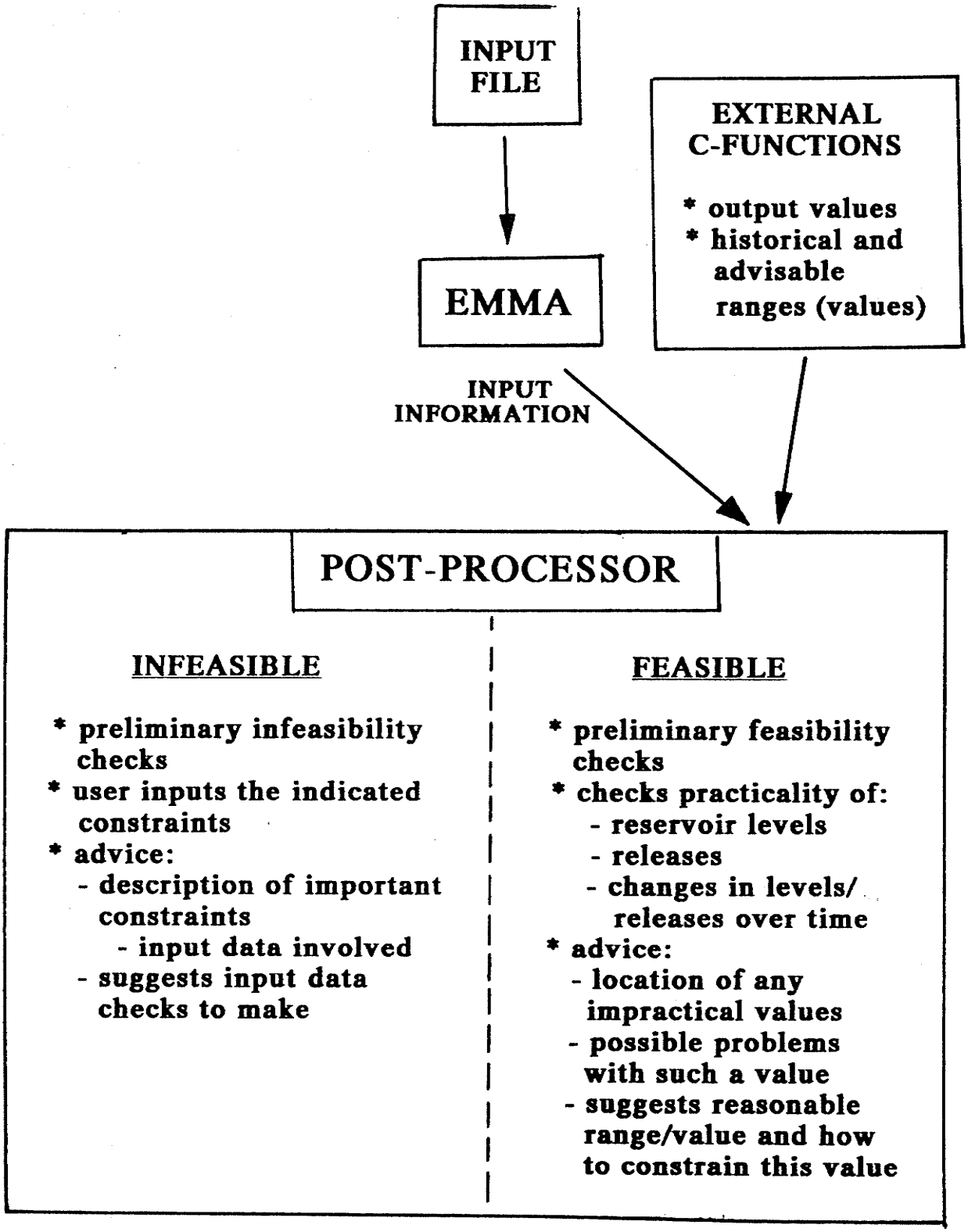

Figure 7. The post-processor to EMMA.

Another rule created through personal experience, was one relating to a possible error that should automatically be noticed by EMMA, in a routine that conducts some initial data checks. EMMA will not continue execution if one of these checks fails. One obvious error seemed to slip through this analysis. Thus, a solution could be deemed feasible, even though a data error was present. The user is warned about this possible situation. Finally, the expert advised that after a feasible run, the user should ensure the primal and dual solutions are the same (a requirement for linear programming). These preliminary feasibility checks are conducted in file **output.ind** of the post-processor (Appendix B).

From his many years of working with the Manitoba Hydro power system and EMMA, the RES Engineer has developed a feel for the values of certain problem variables. His expertise is more directly connected with the hydraulic network, so the output values he selected to be checked were releases and levels. His 'feel', for what reasonable values for these variables are, is derived from what he has seen take place in the past. Thus, in the prototype system, the user is informed if any releases or levels have fallen outside of historically observed limits. A value outside this range may pose a problem. If a release is unexpectedly high or low, the user is warned, as this generating station may never have been run at this level before, and possibly for good reason. Again, artificial (i.e., made up) historical limits have been stored in C functions. These checks are conducted in file **release.ind** of the post-processor.

A further check is conducted on the changes in reservoir releases specified to take place over timestep intervals. For mechanical reasons, a large change in power generation must be accomplished slowly, over a

reasonable period of time. The user is warned if a specified change exceeds the historical maximum release change, for this station over such a time interval. The user is advised on the possibility of constraining a significant change to a more reasonable value. Fictitious historical maximums are stored in C functions, and the release change checks are conducted in file **changes.ind**.

In a similar fashion, checks are made on the practicality of lake levels and level changes called for in a feasible solution. Again, the expert's advice was to use historically observed conditions as checking points. Reservoir levels are checked against historical maximums and minimums, and the user is advised of the location of any unreasonable values and the historical range. According to the expert's suggestion, the user is told how to constrain these values, if so desired, to the historical values. Fictitious limits have been stored in C functions. One can refer to file **level.ind** of the post-processor for these checks.

The level changes called for in the solution are compared with the historically observed maximum change on a particular lake. The user is given the location of any unreasonably large change, and warned about the disruptions to communities such a change might cause. This warning advice was elicited through discussions with the expert. The advice includes a suggestion on how to constrain this change. C functions contain made-up maximum change values for the lakes (in the Laurie River system). File **level_changes.ind** of the post-processor contains this information.

An example rule is presented for checking that powerhouse releases have not exceeded the practical limits.

EXAMPLE RULE:

```
IF (powerhouse_flow > max_ph_flow) IS

"T": (s_advise "WARNING!
                Scheduled release exceeds historical
                maximum.
                Station has never been run at this
                level before!
                Can constrain maximum flow through
                powerhouse to be this historical
                value.";

        print generating_station
                timestep
                scheduled_release
                maximum_advisable)
```

While the knowledge bases previously described contain, almost exclusively, information and expertise derived through discussions with the expert, this is less true of the portion of the post-processor dealing with infeasibilities. In this case rules were generated by the knowledge engineers through a combination of discussion and experimentation, which will be described later.

Running EMMA a user may be faced with no feasible solution possible for a planning problem and request advice on how to deal with this situation. The post-processor assumes that the infeasibility is due to a data entry error. When a solution is infeasible, EMMA produces an output report that underlines all the constraints and decision variables involved in the infeasibility. Through many years of experience, an expert can more efficiently deduce the likely cause of the infeasibility given the variables and constraints involved.

Prior to conducting an analysis of the cause of the infeasibility, the prototype system performs some initial infeasibility checks. As

previously mentioned in connection with the feasible case, the user is asked to check for an error message at the top of an output report, and is told to correct this error and rerun the program. Secondly, the user is informed of a bug in the EMMA program. One can recognize if EMMA ran into this bug by checking if a peaking constraint has been created when it should not have been. If this is the case, the problem should be run again. Rules were also created to advise the user in the case where the previous run was feasible and only a small number of data changes were made. These changes could be checked and, if correct, then made one at a time to discover which one is causing the problem. The initial infeasibility advice is generated in file **prelim.ind** of the post-processor.

As previously stated, the rules developed for handing out advice regarding the cause of an infeasibility are mainly the result of experimentation. Discussions with the expert helped to clarify certain issues. Following the expert's advice, EMMA was run numerous times, changing individual data items in order to create infeasibilities, then the constraints and variables indicated by EMMA were recorded. Changes were made to virtually every data item related to lakes, hydro generating plants, thermal plants, inflows, control structures, natural outflows, crew data, and energy grouping data, for the Laurie River system.

The purpose of this experimentation was to develop a link between the input value change made and the resulting list of indicated constraints and variables. No obvious links could be established. Often times, the magnitude of the change had an effect on the contents of this list. To supplement the knowledge gained through experimentation,

the RES Engineer was observed as he dealt with several concocted infeasibility cases. His approach was, for the most part, trial and error. However, certain important observations could be made.

First, he began his analysis by looking at the indicated constraints only. Secondly, he sorted through them in a specific order, starting with the ones he deemed most important. Thirdly, he referred to the fact that certain indicated constraints were not really important to the analysis, they were simply an offspring of another problem. Finally, he mentioned that it is important if certain constraints (flow continuity, and supply and demand) are not indicated. The prototype system reflects these factors.

The prototype also reflects the knowledge that was generated by the process of artificially creating infeasibilities. Through experimentation, it was only possible to establish general guidelines, as opposed to concrete rules, for dealing with infeasibilities. The prototype system acts as more of an information source, offering descriptions of constraints and the input data involved with them. The following paragraphs describe the contents of the infeasibility portion of the post-processor.

Due to the enormous number of variables that could be indicated in an infeasibility and the expert's checking of constraints first, the analysis of an infeasibility is restricted to the indicated constraints. Thus, the user is asked about the constraints EMMA says are involved in the infeasibility. This input process is conducted by file **constraints.ind**.

Based on this information, the user is provided with a description of the important constraints on this list and the input variables involved with each. The user is also given suggestions on which of these input

data values should be checked (i.e., which is the most likely cause of the infeasibility). The important constraints were identified through discussions with the expert. The following is a list of the constraints the prototype system considers in its analysis:

- hydro production;
- import fraction limit;
- generating station flow bound;
- control structure flow bound;
- grouping constraint (strip);
- maintenance constraints - crew scheduling, required maintenance and crew availability;
- energy supply and demand;
- flow continuity;
- tieline total load.

The order they are checked reflects the results of the experimental runs. Input data checks are first made on constraints that rarely appeared in the indicated list, and were, thus, more of an indicator of the proper input data to check. The analysis is initiated in file **analysis.ind** of the post-processor, however, other files are called which contain constraint descriptions and advice.

Because it was not possible to establish any direct link between the input data change made and the resulting pattern of indicated variables and constraints, several pieces of advice are given to a user. All the important constraints will be described and a number of input data checks will be suggested. It may also be significant if the flow continuity or

supply and demand constraints are not involved, and the user is informed if this is the case. At the end of the analysis, if none of the suggested input data values is the culprit, a possibility is that there simply are not enough resources available to produce the necessary power. The user is then advised on how to create a dummy power plant to ensure demands are met, so a feasible solution is reached, and, therefore, the resulting output reports can be studied. If a solution is infeasible, the resulting output reports will contain a lot of 'garbage', making it impossible to elicit much information from them. An example rule is presented for the case when a hydro production constraint has been indicated.

EXAMPLE RULE:

```
IF (hydro_production_constraint) IS

"yes": (production _constraint_description;
        s_advise "Check for an error in the:
                  MINIMUM CAPACITY REQUIREMENT
                  of the power plant indicated by the
                  constraint.";
        import_fraction_check)

ELSE (null, import_fraction_check)
```

The post-processor contains approximately 50 rules, some of which are used to direct the processing, as in the pre-processor. The system obtains some of the necessary information about the problem structure by accessing the input values through EMMA's own data structures. This prototype, however, is not linked to EMMA on the output side, so it cannot access the information stored for the production of the output reports. For this reason, the user must be questioned directly about the constraints that have been indicated in the case of an infeasibility. As well, output values had to be stored and accessed through C functions.

Again, the historical, or advisable, values are stored in C functions which are integrated into the RuleMaster modules. A listing of the modules in this prototype system are provided in Appendix B.

7.1.3 Current state of the prototype systems

As they stand at this moment, neither the pre-processor nor the post-processor is of any significant practical value to an inexperienced user of EMMA. The prototype systems reflect more the problem understanding of the knowledge engineers, as opposed to that of the expert. Much work would be needed to bring these prototype systems up to the expert performance level. With the new, hierarchical, detailed model (Figure 5) of the RES Engineer's use of the EMMA computer model, a more suitable top-down approach to development is possible. Thus, much greater success in the knowledge acquisition phase, and, in turn, more 'expert-like' systems are expected in the next phase of the project. Besides, many of the computer and tool problems have been ironed out. The development of the prototypes were, however, of significant value to the knowledge engineers, as they learned a great deal about the domain, and also about expert systems.

An expert system developer is warned against having unrealistic hopes of creating truly expert systems, without devoting a great deal of time and effort. Many experienced developers cite long development times as one of the drawbacks of expert system projects, sometimes pointing to a figure of several man-years as a general rule of thumb for developing a full scale expert system [2,17,19]. Of course, the actual development time horizon depends on the size of the project, the

domain, and the number and quality of project team members involved. The Manitoba Hydro project was very large, the domain of water resources is extremely complex, and the development team was relatively small and inexperienced. The trick is to start small with a relatively simple problem and, therefore, be encouraged, as opposed to discouraged, through the development process. One article warns against investigating tasks that require more than a few hours of expertise to perform, and, therefore, the knowledge is too complicated and unbounded [6]. Depending on the motivation for running EMMA, the problem formulation and output validation can take hours to days. The next section describes a more successful attempt at creating a prototype expert system, and possible reasons why it was more successful. To gain initial experience with expert system tools and techniques, development teams might begin by investigating a problem similar to the one presented below. This background experience will make them more prepared to attempt knowledge acquisition from a human expert.

7.2 Preliminary dam design case study

7.2.1 Description of the problem domain

A requirement of a graduate level course is the development of a prototype expert system, in any domain. Previous work on the preliminary design of a dam indicated the suitability of much of this area of expertise to expert system techniques. Due to the uncertainty involved in the hydraulic, geotechnical and structural considerations, assumptions, or reliance on empirical formulas or heuristics, may be needed. In fact, the reference source for the course contained several

design "rules of thumb" developed by an experienced hydraulics engineer.

For a small expert systems project, success will also depend on the availability of expertise. Many students simply chose a problem in which they, themselves, could act as expert. The preliminary dam design domain was too large for such a project, but a report was located, produced by the National Research Council (NRC), on flood and earthquake design issues [40]. The report was a compilation of the results of an extensive study into current design practices in dam engineering. Unfortunately, the NRC was unable to reach a consensus for recommending specific criteria for flood or earthquake design. The information they had gathered was, nonetheless, important and potentially useful. No doubt, designers would benefit from having access to a permanent collection of current design practices to consult.

7.2.2 Development of a prototype system

The actual development of rules from the NRC report was a simple and straightforward process, as compared with the Manitoba Hydro project. The domain knowledge of dam design experts had been elicited, interpreted and organized by the NRC. The only remaining task was to transfer this information to an expert system. On the other hand, the Manitoba Hydro project involved not only human communication with an expert, but a certain level of domain and problem understanding. The transfer of the preliminary dam design information could be, for the most part, performed without extensive domain knowledge.

The information had been organized, within the NRC report, whereby design advice was related to specific dam parameters. The selection of a

spillway design flood had been related to the size and hazard classifications of a dam. The size class is determined from the values of dam height and volume, while the hazard class is dependent on damage estimates in the event of a failure. Estimates of the loss of life and economic loss are given qualitative values (eg. minimal, appreciable, excessive), based on the extent of development in the surrounding area. Thus, from these values the prototype expert system determines the size and hazard classifications and then produces the appropriate list of spillway design floods used in practice for a given combination of classes. To demonstrate the ease of rule construction Table 1 was taken from the NRC report [40] and possible rules, regarding size classifications, are presented below it.

Table 1. Size classifications for dams

CATEGORY Size of Dam *	IMPOUNDMENT (ac-ft)	HEIGHT OF DAM (ft)
small	50 to 1,000	25 to 40
intermediate	1,000 to 50,000	40 to 100
large	over 50,000	over 100

* Criterion that places project in largest category governs

EXAMPLE RULES:

RULE 1: IF (volume > 50000) OR (height > 100)
THEN size = large

RULE 2: IF (1000 < volume <= 50000)
OR (40 < height <= 100)
THEN size = intermediate

RULE 3: IF (volume <= 1000) AND (height <= 40)
THEN size = small

The prototype system does not pinpoint one particular value for the safety evaluation flood, unless it is a high hazard dam, as the NRC only made a recommendation for this case. Extending the prototype could involve a thorough explanation of the analysis procedures required to derive such values as the probable maximum flood, which are used for design purposes.

On the other hand, the table of current practices in earthquake design was not as helpful. There was no indication as to when a particular analysis technique should be employed based on some definable project characteristics, such as dam size. However, a simple table was provided for use in pseudostatic analyses. Minimum earthquake design coefficients were directly related to the seismic zone classification of the area. In the conclusions of the report, suggested design considerations were dependent on the type of dam (earth or concrete), and some measure of local seismicity, either seismic zone (concrete dams) or peak ground acceleration (earth dams). Finally, the selection of a safety evaluation earthquake is related only to the hazard class. The NRC made a strict recommendation only for high hazard dams. Again, the prototype designed around this information could be extended to offer further clarifications on the suggested analysis procedures and design measures.

Due to the fact that the knowledge base was derived from a static source (the NRC report), no formal knowledge acquisition was performed.

The presentation of some of their findings regarding flood and earthquake design criteria allowed for quick and easy transfer to rules. Classifications and design advice depend on the values for a relatively small number of dam project parameters. The system presently contains 30 rules and took only a few days to develop. It was developed within VP-Expert, a PC-based expert system development tool. A complete listing of this prototype is contained in Appendix C.

Perhaps, the overall usefulness of the prototype system could be greatly enhanced by including the expertise of local designers (i.e., how do the design practices of Canadian federal and provincial agencies differ?). Knowledge acquisition will then become an issue. As it happened, the NRC had performed the knowledge acquisition, and also organized the information. It might prove helpful to contact them and find out how they approached the problem of eliciting knowledge from experienced dam engineers. The prototype could also be extended so it is broader, dealing with a larger number of design issues, such as wind or ice.

The preliminary dam design domain appears to be extremely compatible with the development of expert systems. This personal view is further validated by reports of similar projects in the literature. In one case an expert system was developed to assist in assessing the safety of an existing dam, relying on qualitative field inspection observations and more quantitative safety factor calculations and comparisons [23]. Another expert system was designed as a documentation of the provisions from a tentative seismic design code [1]. Its purpose is to help designers select parameters for seismic design. Such attempts are useful as they result in the documentation of current design practices. A designer would benefit

from knowing what his, perhaps more experienced, colleagues are doing and what has been found to work, or not work, in the past.

CHAPTER 8.

ENGINEERING KNOWLEDGE

8.1 Engineering today

One objective of this thesis was to examine existing expertise in the water resources domain, in connection with expert systems. The following is a discussion of this area based on personal experience and ideas presented in the literature.

The recent Fourth Canadian Seminar on Systems Theory for the Civil Engineer included a session on expert systems and decision support systems. One engineer remarked, during the discussion period, about the lack of documentation done by practicing engineers. He suggested that if engineers were to religiously document their work there would be no need for expert systems. This is not, however, common. Since engineers are not in the habit of writing detailed descriptions of their work, little is known, even by the experts, about their domain knowledge and problem solving processes, or how they have evolved. A great deal could be learned if an engineer's gradual climb to the expert problem solving level was documented. Instead, a knowledge engineer is sent in, after the engineer has reached this performance level, to now document his expertise in the form of an expert system.

People don't have to be reminded of the technological changes in society that have taken place in recent years. As well, economic issues have begun to dominate all levels of decision making. There is added pressure to produce more with fewer resources, people included. It is becoming increasingly more difficult for inexperienced engineers to gain

adequate training. In fact, it has been observed that the engineering profession is steering away from long and detailed apprenticeship or training programs [9]. This tendency was noted by another participant in the systems conference. In his article, Russell [49] admits that many offices do not allow graduated engineers the opportunity to work directly under the supervision of an experienced engineer. As a result, "their crucially important professional judgement, has to be picked up in an ad-hoc manner" [49].

Less time may be spent on training and teaching due to the pressure of today's society. This is true for most professions, including water resources engineering. Many have been convinced that expert systems could be a partial answer to this problem. Certainly, these systems could prove quite useful. An inexperienced engineer may feel more comfortable interacting initially with a patient, consistent, non-judgmental expert system. With the confidence and expertise acquired in this manner, he can then perform better, and, ultimately, communication and learning from the expert are enhanced.

8.2 The computer - friend or foe?

Hayes-Roth [27] describes the traditional approach to gaining expertise in a domain through the following passage:

> "For nearly 500 years books have been the primary means of retaining knowledge and transmitting it to humans. To achieve excellence in a profession, humans have studied, interpreted, and memorized these books; apprenticed and trained with someone who could clarify and illustrate the book's principles; then practiced for years and learned practical rules from experience."

He goes on to suggest that in many technically advanced fields, the creation of knowledge has outpaced the speed at which it can be disseminated, understood, and used. He points the finger at the computer for creating the need to enhance knowledge distribution. It is also the computer that provides the opportunity to do so, through the development of knowledge-based systems.

Engineering science is indeed advancing, leading to the need for radical specialization and, thus, the scarcity of expertise. The computer has allowed for the creation of more complicated models, and the collection and analysis of immense amounts of data - so much so, that the human brain could not possibly process all the available information. This then leads to a need to select and implement only the most relevant data. Still, it remains a problem that the "right" data is not always available, nor at the desired level of certainty. As a result some assumptions and speculations will be needed.

In fact, civil engineering problems can be categorized, according to Alim [1], by their:

- inherent imprecision;
- the paucity and incompleteness of data;
- fuzzy decision processes;
- heavy reliance on expert views, which are themselves vague and imprecise.

Designs or solutions must be robust enough to withstand the uncertainties that incomplete knowledge causes [29]. When representing existing expertise in a water resources domain, uncertainty and

imprecision are vital issues. This may mean relying on confidence factors or simply providing the user with several suggestions or pieces of advice, and leaving the final decision to him.

Two prominent critics of expert systems pointed to the uncertainty problem as one of the major shortcomings of the expert system approach. In fact, the Dreyfus brothers [16] state that expert systems are no more able to deal with uncertain data than they were a few years ago. The enthusiastic support for expert system development is viewed as a "misguided effort" [16].

Computers are limited not only in their ability to represent uncertainty, but, also, in their ability to: learn; represent, store, and manipulate knowledge; reason in time and space; and understand their knowledge. These computer hardware related limitations are responsible for many of the limitations of expert systems discussed in section 6.7.

Perhaps, it would be better to look at what computers do have to offer, as opposed to what they are lacking. Through the design of hybrid decision support systems [21,42,44,50], the role of the computer can indeed be extended. Expert system technology can be combined with computer graphics, systems analysis techniques, engineering expertise, and databases. The resulting hybrid system is said to produce a computer which can act as a mediator and translator between experts and other affected parties (public, politicians, etc.), or, in other words, science and policy [21]. The computer then becomes a "vehicle for communication, learning and experimentation" [21].

8.3 Possible roles of expert systems

One's opinion of expert systems may depend on how their possible role is envisioned. Certainly knowledge-based systems created to act as checklists or alternative information sources would be a helpful, and noncontroversial application. In other words, they could represent the theoretical knowledge currently found in textbooks and codes. The debate heats up when people speak of 'expert' systems, which are to act as consultants for non-specialists, or training/teaching tools. Often times, these systems will have to include many human opinions and judgments.

One can further distinguish between the idea of expert systems as problem solvers and expert systems as advisors [11]. The former classification calls for the systems to obtain a concrete, communicable solution to a problem. In many water resources domains, complexity and uncertainty prohibit the design of such systems. The role of these systems becomes, more appropriately, that of advice giver. The focus shifts away from the solution towards more understanding of the problem content. With either scenario, an increase in knowledge about a problem can simply result from the expert system's ability to examine all possible combinations or quickly induce rules from an example table. Thus, they are potentially capable of producing solutions or approaches to problems that were previously overlooked by human experts.

At this time, humans appear more capable of dealing with many, apparently extraneous, factors, recognizing their relevancy, and dealing with uncertainty. An expert system should simply assist in selecting, gathering, and interpreting information relevant to a particular problem. The user will be required to integrate this information and judge its

relative impact on the outcome. These thoughts will be further developed in the following sections which introduce the idea of categorizing engineering knowledge.

8.4 Classifying water resources engineering expertise

In order to discuss engineering knowledge in the context of expert system development, it is necessary to derive suitable classifications for this knowledge. One can approach the problem from several directions. Engineering knowledge can be compartmentalized by examining it in accordance with the type of problem involved, the types of knowledge used, or the structure of that knowledge.

8.4.1 According to problem type

We shall begin with the simplest categorization strategy, according to the problem type. The following list of problem categories was taken from Allen et al. [2], although it has appeared, in one form or another, in several other references. The descriptions include potential design issues with respect to expert systems.

(a) **Analysis assistance**:

- intelligent offering of advice to a user of complex computer software;
- principal development issue is working with incomplete and sometimes erroneous data;
- eg. parameter selection for the SWMM model [3].

(b) **Diagnosis and Interpretation**:

- diagnosis -> process of fault finding in a system;
- interpretation -> intelligent collection of data;
- development issues include -> partial, contradictory or unreliable data;
- eg. interpretation of the existing condition of a water body, or failure diagnosis in a dam.

(c) **Design**:

- creation of a system or object satisfying given specifications;
- ensure goal satisfaction without unnecessary resource consumption or constraint violation, and establish priorities for conflicting goals;
- design issues include -> consequences of design decisions cannot be predicted until the design has progressed considerably, hierarchy of subproblems almost always a necessity, spatial reasoning required;
- eg. preliminary dam design.

(d) **Monitoring and Control**:

- monitoring -> process of continuous or intermittent interpretation of signals;
- control -> adjusting a system to the signals (feedback) received;
- design issues include -> data may be partial, contradictory or unreliable, and real-time responses may be desired;
- eg. operation of a sewage treatment plant.

(e) **Planning**:

- creation of a program of actions to achieve a set of goals;
- design issues include -> those associated with design, constraints and goals must be accommodated, and occasionally relaxed, in an acceptable manner, and solution requires temporal reasoning;
- eg. planning the short-term or long-term operation of the Manitoba Hydro integrated power system.

In two articles dealing with engineering and expert systems, a logical grouping of these problem categories was proposed [2,18].

1) **Derivation-type problems**:

- analysis assistance, diagnosis and interpretation, monitoring and control;
- involve the ability to obtain data, reason with it, and interpret it to draw conclusions and suggest alternatives;
- require experienced, reasoned, and, often, heuristic judgement.

This type of problem solving could be assisted through rule-based expert systems. In many cases, an expert system could act as a problem solver, actually guiding the user through the analysis and arriving at a solution. The principal issues facing an expert system developer are the need for strategic (long-term) reasoning or tentative (short-term) reasoning, data is time-varying and may be incomplete or erroneous.

2) **Formation-type problems:**

- design, planning;
- involve the creation of higher-level objects from integrating lower-level classes of objects.

Formation-type problems present a greater challenge to developers of expert systems. Often, the number of possible solutions to a planning or design problem exceed tremendously the number of reasonable solutions. As well, formation-type problems call for tentative reasoning. In most situations involving this type of problem, it would be unrealistic to assume an expert system could be designed as a problem solver. The system should instead provide a design checklist, or act as an information source on the various problem components, but not be a decision-maker. The idea should be to assist a user in gathering and interpreting the information necessary to intelligently solve a problem.

8.4.2 According to knowledge type

Water resources engineering tasks rely on the compilation and assessment of information from a variety of sources [30], some of which will be easier to document than others. This leads into a discussion of another approach to categorizing engineering knowledge. One can differentiate between the various subdisciplines in which a water resources engineer will possess specific, problem-related information.

All water resources engineers will have general background knowledge about technical issues, analysis techniques, and the theoretical behaviour of the physical systems they are dealing with. For example, in the dam design domain, an engineer will have some level of understanding

of hydraulic, structural, and geotechnical design issues. An experienced engineer develops his knowledge in areas of his formal, educational training that augment his problem solving abilities. He has taken this background knowledge, and compacted and organized it so he can mobilize it more skillfully. The ability to correctly modify and compile his knowledge base comes only through experience. The result is a combination of knowledge from many areas of study and heuristic rules referring to their use.

The uncertainty inherent in the water resources domain has been referred to quite frequently. Due to the stochasticity of water-related processes, such as rainfall or streamflow, an engineer will require a working knowledge of reliability and risk analysis procedures. Conducting studies of risk or reliability calls for knowledge of aspects of economics, statistics, sociology, political processes, and any other discipline that will play a role in the final decision-making process. Since uncertainty exists, a plan is derived that achieves the desired level of reliability (risk). To do so, a water resources engineer must gather all relevant information from the involved specialty areas.

There is a great deal of subjective judgement in engineering decision-making, "in predicting not only economic or social impacts of alternative decisions, but physical, biological or chemical impacts as well" [32]. Mathematical modelling should generally be restricted to elements that can be represented with some level of credibility. Other factors can be brought in, and considered alongside the direct modelling results. This is the point where a more intelligent, manageable, useful, and reliable interface, i.e., an expert system, becomes an asset to an inexperienced

engineer. Experience is required to judge the expected behaviour of a water system. The exact behaviour can never be predicted, but given certain problem characteristics an expert can make a skilled and informed prediction. An inexperienced engineer, with his limited knowledge of technical and reliability factors, is not in a position to make such a judgement. An ill-informed or unskilled engineer is more likely to make incorrect, and potentially dangerous, judgments.

The way physical systems are presented in textbooks cannot always be a true indication of their actual behaviour, when taking into account the stochastic nature of processes such as streamflow or precipitation. The idea then is to plan for the future, and plan for the unexpected, within some reasonable bounds. Reliability or risk levels are factors no one likes to explicitly state. However, access to this information is vital to decision-making. Textbooks explain the theories and techniques involved in reliability or risk analysis, but it is another thing altogether to apply these intelligently to a real system, ensuring a safe and reliable solution at minimum cost.

The type of knowledge to be elicited will have an impact on the development of expert systems. Much of the general background knowledge of an engineer can likely be represented in an expert system. However, with issues such as risk and reliability, which involve qualitative knowledge from many fields, experts are wary of widespread circulation of their approaches. This is especially true if they believe an inexperienced user can potentially misinterpret or misuse them. A knowledge engineer must be aware of this reluctancy and convince the engineer his expertise will be part of an advice system, not a decision-

making system. The user must also be aware of the uncertainty involved, perhaps given access to several expert opinions, or given acceptable ranges for safety factors, probabilities, or other measures of reliability, as opposed to a single value.

8.4.3 According to knowledge structure and origin

In their article describing expert systems in river basin management, Jenkins and Jowitt [30] refer to the three most prominent problem-solving assets of an expert: **archived data**; **theoretical knowledge**; and **operational expertise**. Knowledge engineers, or anyone attempting to capture and understand the workings of an expert's mind, must elicit knowledge of the connection between these three problem-solving assets. One must determine not only what is known, but more critical is how and when it is applied [35]. Engineering problem solving is achieved through a combined use of the three aforementioned knowledge sources. Archived data and theoretical knowledge are possessions of any engineer. It is, however, the operational expertise that ultimately links everything together.

8.4.3.1 Data analysis

Decision making in the water resources domain can range from a very structured exercise to a very unstructured one. Structured decisions often involve a more data intensive approach, while in unstructured decision making, human judgement is the key component [41]. Most water resources problems will involve aspects of both types of decision making processes.

In the Manitoba Hydro case study, the expert's knowledge came on three distinguishable levels: **data analysis**; **mathematical modelling**; and **operational experience**. Firstly, a major responsibility of the RES Engineer corresponded to conducting **data analyses**. Using historical data, (i.e., knowledge of the past behaviour of the system), the expert is able to recognize unexpected, or improbable, values. To correctly run a Manitoba Hydro planning problem on EMMA, a user will have to know where to get all the necessary data, how best to analyze it, and what form is needed by the model.

Data analysis can be viewed as a fairly structured aspect of an engineering problem solving process. In water resources data analysis, you are asking questions such as [30]:

- how representative is the data? (temporally, spatially, etc.)
- is the data correct, and how accurate is it?
- can any data be justifiably excluded? (meteorological or operational phenomena, etc.)
- what is the statistical significance of the results?

An expert will rely heavily on what he has seen take place in the past to determine how correct or representative the data are. Standard statistical techniques can be employed to aid in judging the significance, accuracy, and reliability of the data. An expert can take advantage of the rules of thumb often listed in textbooks, which were developed by another expert, and combine these with his own experience. As a result, much of the data analysis problem component can be represented quite closely through a rule-base that guides the user as to what tests to

perform and when, and how to interpret the results. In fact, because there is a relatively small judgement aspect, the approach of the expert can be encoded and a user may be able to conduct the analysis with little input from an expert. This is assuming the expert system is equipped with good help and explanation facilities. Certainly, there will be portions of the problem solving that are more experience oriented and here a system can make suggestions, but the final decision calls for an expert's judgment.

A main ingredient of the Manitoba Hydro prototype systems was data analysis. It was only necessary to identify: the techniques applied in connection with specific data; the historical data used for comparison purposes; the methods used to judge the significance of unexpected data values; and the methods suggested for dealing with these values.

8.4.3.2 Mathematical modelling knowledge

Data, in itself, is of little use unless the information it contains about the behaviour of a physical system is elicited through some modelling technique. Thus, the second asset an expert problem solver has is a storehouse of **theoretical knowledge**. This knowledge base usually includes: technical aspects of the physical system and its components; factors that might influence data; and compatible analytical or modelling techniques. "Modelling", in this context, is a broad term encompassing the use of: formulas; procedures accepted in practice; or complicated computer algorithms.

As mentioned in the introductory portion of this thesis, systems analysis techniques are often inadequate when it comes to modelling

subjective elements of problem solving. To compensate, modelers may have the tendency to increase the apparent elegance and complexity of their models. Engineers speak of the communication gap existing between the model builders and those using the models in practice. Besides engineers, other parties with a vested interest in water resources decision-making, such as politicians or the public, would like to be able to understand the model [28,32]. It is, in many cases, impossible to develop a complete modelling theory for a problem being studied. An important aspect of water resources engineering then stems from the creation and use of models. Model development requires skill and expertise, but so does applying them in practice.

To reiterate, an expert will have developed connections between the data available, his theoretical knowledge, and his operational experience. The choice of a modelling approach depends on knowledge of the available techniques. Experience will help an expert match the characteristics of a given problem with an appropriate analysis or modelling technique. Matching relies on observations of which techniques have enhanced the design, planning or operation of a given system, and which have been found to be of lesser value for a particular problem. With further experiences and increased awareness, these heuristic rules may be modified. Textbook authors often include suggestions or rules of thumb they, or other experienced personnel, have developed in connection with a certain theory or technique. An expert system can capture the logic behind the analysis or model selection process. Equally important to the inexperienced engineer is advice on why some of the other techniques are not as applicable.

The RES Engineer at Manitoba Hydro relies on his combined knowledge of the hydraulic and electric systems, and mathematical modelling to solve his problem. The depth of an engineer's mathematical modelling expertise will depend on the size and complexity of his particular problem. This expert has extensive knowledge of general linear programming theory that is problem independent. For example, he understands the simplex method used to determine the optimal solution, along with the meaning of slack variable and dual prices. He also has developed an understanding of the EMMA modelling tool as it applies to modelling hydraulic and electric systems. For example, he knows what constraints can be created and the format of an input file. He also possesses knowledge about the concepts of feasibility and infeasibility in optimization techniques, as related to an integrated power system. Much of this knowledge can be found in operations research textbooks and the EMMA manual, for anyone to study. Where the situation gets trickier is in combining this knowledge and applying it to a particular integrated power system, to obtain not only a feasible solution, but a practical, reliable, socially and environmentally conscious plan for long-term or short-term operations.

8.4.3.3 Operational experience

The preceding paragraphs have introduced the third and final asset of an expert engineer, that being his **operational experience**. This is what ultimately distinguishes a novice problem solver from an expert. One author put it this way: when he knows how to apply the knowledge he has learned in school, he becomes an expert [35]. An engineer

incrementally builds up his expertise through experience. This can involve integrating new facts, adjusting reasoning procedures, or modifying, adding, or deleting heuristics. Accomplishing this useful compilation of knowledge bits requires that an engineer take notice of failures and successes, and the differences and similarities between these cases. He, thus, learns how to deal effectively with not only the expected case, but unexpected or unusual conditions. An extremely important aspect of knowledge engineering is then to recognize when several parts of different problems are conceptually related, looking for potential general and specific heuristic rules.

A water resources engineer learns by doing and comparing. He applies his knowledge to a problem scenario based on what has worked in a similar situation in the past, in hopes of duplicating the results. If the results obtained are better or worse than expected, he determines the differences in the initial problem states, and tries to find the one(s) that was responsible. This new knowledge is stored away in some heuristic form, so that if this new situation arises again he will have an idea of how to adjust his decision-making. A knowledge engineer takes a similar approach when testing a conceptual model or a prototype system of an expert's problem solving process. Just as in systems analysis, one compares the actual behaviour of the system with the expected [24].

Water resources engineering experts demonstrate expertise during the acts of "clarifying the problem, suggesting the kinds of procedures to use, judging the reliability of facts and deciding whether a solution is reasonable" [17]. Expertise in water resources planning and management is, therefore, equally dependent on experience as it is on theoretical

knowledge. This is certainly the case in the reservoir management domain at Manitoba Hydro. There is a standard set-up for the hydraulic and electric systems, so portions of the input file rarely change even when the motivation behind using EMMA changes. The RES Engineer is most concerned with the data that contains some level of uncertainty, for example, forecasted loads or streamflow. His long-term operating plan must include provisions should the unexpected occur. Therefore, his analysis includes a study of a range of scenarios, including the expected case and a worst-case.

From past experience, the RES Engineer has some idea as to how the system can be best operated under a given set of assumed conditions. Through the knowledge acquisition process, it was discovered that this operational expertise involved issues such as:

- conditions to be assumed for a drought situation (eg. use 5% inflow forecasts) and assumptions regarding a severe winter;
- predicting the behaviour of operators who share resources with Manitoba Hydro (eg. what level can you expect in a lake regulated by some outside board?);
- incorporating concerns of lake users (eg. minimizing lake level fluctuations or ensuring minimum releases for fish);
- planning for the unexpected, within reasonable bounds (eg. assuming the breakdown of some units in the power generation system).

His one overriding objective is, as he puts it, to "keep the lights on" in Manitoba. Much of how this expert incorporates all these concerns is

unclear at this stage. Slowly, through regular discussions, these issues and his reactions to them are becoming clearer.

It is the knowledge that originates from experience that complicates expert system development. Problem solving tasks involving a great deal of operational expertise will be more difficult to model. An engineer will make extensive use of experience-oriented knowledge when dealing with environmental, social, safety, reliability or political issues, which arise at the final decision-making stage.

8.5 Rule development

As an indication of the form of the RES Engineer's heuristic rules and the knowledge acquisition procedure followed, the development of one of the prototype system's rules will be presented. Rule development is a time consuming task requiring repeated reference back to the expert for further clarifications. The rule discussed to demonstrate this iterative process is the same one that was presented in the description of the Manitoba Hydro pre-processor prototype system.

The expert initially advised the knowledge engineers about the importance of checking such input values as inflows, starting reservoir levels, and production coefficients. In the case of the production coefficients, the knowledge engineers initially consulted the EMMA manual for information. The next discussion clarified the reason for checking this parameter. Initial estimates can affect the convergence of the iteration procedure EMMA uses to solve the non-linear energy production equation. The rule initially had the following form:

IF (production_coeff is outside_reasonable_bounds)

THEN warn user of convergence problem

It was then necessary to ask the expert about the suggested reasonable bounds on this parameter. The knowledge engineers were informed that estimates are usually based on the normal operating head of a plant, but the practical limits should correspond to the range limits of the corresponding reservoir storage curve. The calculation procedure was demonstrated for one lake. Thus it was a rather straightforward process to determine these bounds.

IF (input_production_coeff > practical_maximum)

OR (input_production_coeff < practical_minimum)

THEN (give_limits and warn_of_convergence_problem)

This rule can then be easily extended to include a suggested value based on the observed normal operating head. In a later meeting, the expert revealed that even with a fairly bad estimate the system converged quite rapidly.

This rule demonstrates a combination of data analysis (calculation of production coefficients from storage curve), theoretical knowledge (use of production coefficient iterations to pose a linear programming problem), and operational expertise (what a good estimate is for a particular lake and how convergence is affected). The rule was quite easy to develop because the decision was more structured, i.e, data intensive.

To contrast, a failed attempt at eliciting a type of rule for the

prototype systems will also be discussed. The objective function of the optimization procedure in EMMA involves maximizing net benefits in dollars. Therefore, it includes cost coefficients to reflect export and import prices, and costs associated with energy production, which are understandable. However, the objective function also includes cost coefficients associated with storage, flows, and scheduled maintenance, which do not simply reflect direct costs or prices. An attempt was made, through interviews, to determine how these latter coefficients were assigned. Because this assignment ultimately depended on the specific motivation behind an EMMA run (we were not aware of the hierarchical model of motivational problem contexts at this point), the explanations were incomprehensible. Further attempts to clarify this portion of the RES Engineer's expertise failed.

The setting of cost coefficients appears to rely heavily on a mixture of EMMA modelling and system operating experience. Without knowledge of a specific problem context to work within, the expert could not possibly pinpoint the approach he takes to determining some of these cost coefficients. Running the problem many times before, he has developed a good idea of how to assign coefficients so the model behaves as he wishes. For example, relative storage costs will affect the tendency to release water from the various reservoirs. In addition, relative to import and export prices, these coefficients can have an impact on how much of these two resources will be utilized to meet demands. EMMA will try to find the cheapest way of satisfying the energy demand. Thus, it appears that the setting of coefficients related to storage, flows, and scheduled maintenance will depend on what you are

trying to accomplish with a particular EMMA run. For example, in a drought the emphasis may be on keeping water in storage, minimizing flows, and relying more heavily on thermal generation and imports.

CHAPTER 9.

RECOMMENDATIONS

In the following sections, recommendations will be made with regards to water resources engineering expertise and the proposed view of expert systems in the water resources domain. As well, general guidelines are provided regarding the development of expert systems in this domain, and, more specifically, the knowledge acquisition phase of development. These ideas have been developed based on an extensive review of expert system literature, and practical experience on a small scale (preliminary dam design prototype) and on a much larger scale (the Manitoba Hydro case study).

9.1 Water resources expertise

Water resources engineering has its basis in scientific principles. Traditionally, however, this scientific understanding has been limited in its depth. Uncertainty is an important aspect of the water resources domain, due to the inherent stochasticity of processes such as rainfall, or streamflow. As well, water resources engineering problems are typified by their multidisciplinary nature. Decision-makers must combine the results of numerical analysis with political, social and environmental considerations, which, due to their qualitative nature, must be left out of the modelling process. As a result, past experience and expertise become important.

Following a concept presented in an article by Jenkins and Jowitt [30], the suggestion being made is that, for the purpose of expert system

development, existing water resources expertise should be classified according to its structure or origin. An expert will possess knowledge from three different sources, whether it be in the context of planning, design, or operation. **Data** will be accessible reflecting the historical behaviour of the physical water system. In addition, an engineer has extensive **theoretical knowledge**, and, what sets him apart from average workers, a great deal of **operational experience**. Expertise is demonstrated through an ability to combine these three knowledge sources to efficiently and effectively solve a problem.

As exemplified by the work of the RES Engineer at Manitoba Hydro, major tasks involved in solving a water resources problem are: an analysis of the **data**; deriving an acceptable **modelling** approach for the problem; and determining the most feasible and practical **solution**. These tasks are performed whether the motivation is the planning, design or operation of an engineering system. The first activity, data analysis, seems easiest to model in an expert system. Data-intensive tasks are usually relatively structured and rely on a combination of theoretical knowledge of analysis techniques and tools to use in conjunction with the data. There may be portions of operational experience that come into play, as experts can quickly recognize a good approach to eliciting information from data. Difficulty may also arise, for non-experts, when data is missing or uncertain. For the most part, however, experts have developed an almost procedural approach to analyzing data in connection with a specific problem, simplifying the task of rule development.

In the second stage of problem solving, an engineer is faced with the task of finding a suitable method of modelling the physical system

that is involved. Here, the task of expert system development becomes a degree more challenging, as this activity begins to depend more heavily on the past experience of the problem solver. An expert will have developed an idea as to which modelling techniques are most appropriate, and produce the most useful results, given his objectives. As well, he has developed an approach, now virtually subconscious, to representing the physical problem within the chosen model. Many of the heuristic rules governing this process, though second nature to an expert, can likely be elicited through standard knowledge acquisition procedures. This is especially true if tricky issues, such as environmental or social concerns, are left out of the direct modelling process. The ease of knowledge acquisition will depend on the complexity and elegance of the model.

Where the most problems are bound to arise with knowledge acquisition from a water resources expert is at the final decision making point. At this stage, subjective data becomes more of an issue. An expert must rely extensively on past experience, (i.e., operational expertise), to make the necessary skilled and informed judgement. Usually, the expert is unaware of any standard procedure for judging the overall practicality and viability of a solution, including environmental, political, risk-related and social concerns. It becomes a challenge for the knowledge engineer and expert to create adequate heuristics to reflect this process.

Some aspects of the final selection stage of water resources engineering can be modelled effectively in an expert system. Issues related to a theoretical understanding of the modelling approach and interpreting the output data may be modelled through heuristic rules. For example, as with the post-processor for the Manitoba Hydro case study, the practicality of

the solution can be partially judged through comparisons with the past behaviour of this system or similar systems. These subtasks are more data intensive, thus more structured. This discussion will be supplemented in the next section, where some personal views on expert systems will be presented.

9.2 Expert systems in water resources engineering

The general trend in engineering seems to be away from long and detailed apprenticeship or training programs [9]. Thus, enormous benefits could result if inexperienced engineers had access to documentation on problem solving in a particular domain. Expert systems are simply one form this documentation could take. Engineering experts are not used to fully documenting what they do, so a knowledge engineer can assist them in expressing their knowledge in a coherent and explicit form.

An idea expressed in one expert systems article is that engineers might better optimize systems by simply more widely distributing existing expertise [33]. Expert systems should, thus, be viewed as an additional tool for managing the diverse and expansive knowledge requirements of water resources engineering. The emphasis of development must emphasize explanation. This is especially true when uncertain and subjective data is involved.

The main focus of development should be creating **advisory systems**, systems that will act as information sources or guides within a problem domain. A user should be educated through interactions with the expertise captured in the system. These initial rules will allow beginners to accumulate experience [16], but soon they must return to a more

standard knowledge transfer process, through direct interactions with an expert. For example, as Russell [49] explains it is difficult to define the design process in a series of steps. Users might benefit from having access to existing successful designs, a design checklist, or general rules of thumb on component sizing. It may be necessary to leave the final selection and the choice of an acceptable safety factor up to a human expert. Similarly, dealing with certain political, environmental, social or reliability issues may be too difficult to describe through a series of rules. Instead, an expert system might remind a user of the issues that are involved and suggest possible approaches to dealing with them.

Due to the specialization necessary in water resources engineering, expertise in any singular problem domain is scarce. Engineering expert systems, themselves, must, in turn, be very **problem specific**. It is impossible to imagine an expert system that can handle a wide range of water resources problems. As a result, small micro-based expert systems have an important role to play [53]. It is not necessary to tackle large problems to produce noticeable benefits. A good starting point, and an important area of growth for expert system development is the creation of **"intelligent" interfaces** to accompany more standard computer codes. A standard complaint of engineers is the lack of communication between model developers and potential users. This approach was taken in the Manitoba Hydro case study, which centered around the EMMA computer model. Pre- and post-processors aid in data preparation and output analysis. Data-intensive tasks are usually more structured, easing the task of knowledge acquisition. Difficulties arise when subjective data, such as environmental impacts, must be considered.

In the end, it is important for all members of an expert system development team to be aware of the limitations of these systems. They are less likely to be correct, and have a limited knowledge base, when compared to a human expert.

9.3 For the development team

(a) If the development team is relatively inexperienced, start with a small, simple, preferably structured, problem to gain experience with expert system tools and techniques (where tabular information exists, for example, the NRC report on flood and earthquake design criteria [40]).

(b) Begin development of a prototype system immediately. A prototype is useful for knowledge acquisition purposes (expert can criticize system), keeps the expert interested, makes it easier to notice and correct errors, and helps the knowledge engineer learn about the domain.

(c) Spend considerable effort on determining an accurate conceptual model of the expert's entire problem solving process, and choosing a scope for prototype development. Spending time on these activities may ease knowledge acquisition tremendously.

(d) Try to separate the problem into subtasks, so you only deal with a subset of the expert's expertise at any one time, (i.e., modularize the system).

(e) Expect long development times (six months were required for the Manitoba Hydro prototype system and several years are expected for the development of an 'expert' system).

(f) The selection of a knowledge engineer is extremely important. Need someone who will be persistent, ask questions, and continually demonstrate what he has learned. For engineering expert systems, it is suggested that the knowledge engineer be an engineer with previous experience in the domain. A side-benefit is that he is trained, to a degree, in the problem domain. He also must be compatible with the expert. It may be a good idea to involve more than one person in the knowledge acquisition sessions.

(g) Allow the knowledge engineer a training period with development tools and knowledge acquisition.

(h) Rely on incremental development. Start with common problem scenarios, and build in complexity and exceptions. Knowledge is added piece by piece with continual testing of the system.

(i) In the beginning, more emphasis should be placed on collecting and organizing the theoretical knowledge (often available in texts and manuals) and then operational expertise.

(j) Constantly evaluate progress. Don't be afraid to start again.

(k) Investigate the incorporation of uncertainty in the development tool. It may be appropriate to provide several pieces of advice to a user, ranked according to accuracy or importance.

(l) Expert systems must provide advice, therefore, help and explanation facilities are very important.

(m) Large projects require a well qualified computer programmer [14].

9.4 For the knowledge engineer

(a) A co-operative, enthusiastic expert is needed, who is aware of the large time committment required. Educate him about expert systems- what to expect and when.

(b) Familiarize yourself with the domain - concepts and terminology.

(c) Establish not only the knowledge an expert possesses, but how and when it is used.

(d) If qualitative information is required, make sure you understand the expert's view of the terms and make a user aware of this interpretation.

(e) Make knowledge implicit in the rules, explicit to the user, through help or explanation facilities.

(f) Knowledge engineering is not a one-step linear process [18], rather it is an iterative process. Knowledge is elicited piece by piece.

(g) Rules are often the most natural and understandable form of knowledge representation.

(h) Set a regular meeting time with the expert, even if some turn out to be more of coffee-break chats.

(i) Record the sessions, if the expert agrees.

(j) Try to fill in the gaps in your interpretation of the expert's reasoning through continued questioning or criticism of prototype rules.

(k) Procedures used to obtain information from the expert depend on the specific problem and expert involved. Usually, rely on a combination of observing an expert solve a problem, and discussions and questioning to analyze what was done. Rules can be established by working through several problems with the expert and examining the differences and similarities between them and his approach to solving them. Often, initial experience can be gained by playing around with the tools or techniques the expert is using (for example, running EMMA as a means of understanding the problem). Most often, the only way to uncover true expertise is to interpret what was learned in a session by establishing rules or a small prototype system. An expert may find it easier to criticize than say what is needed. As sessions proceed, an idea will be developed as to which approaches work best with the expert.

CHAPTER 10.

CONCLUSIONS

This thesis has focused on the knowledge acquisition phase of expert system development. The objectives were twofold: to discuss the process of knowledge acquisition in the context of engineering expert systems; and to characterize expertise in the water resources problem solving domain. The main sources of information for these discussions were two expert systems projects and an extensive literature review. A simple prototype was developed to assist in the safe design of dams with regards to floods and earthquakes. At the same time, during the first year of a three year project with Manitoba Hydro, prototype systems were developed to accompany a computer modelling tool used in the reservoir and energy management domain.

Whatever the actual prototype systems show, the work undertaken in connection with this thesis has provided ample learning experiences. The problems encountered and the discoveries made, during the development of the 'expert' systems, paralleled those reported in the literature by others involved in expert systems work. It is, indeed, impossible to define a set of procedures to ensure the successful transfer of information from expert to knowledge engineer. Rather, guidelines were produced, with regards to expert system development and knowledge acquisition, specifically tailored to the water resources domain (although they would likely be applicable in many engineering problem domains).

Perhaps, where the greatest benefit of expert systems work comes is in assisting engineers to understand how experts organize and use

their knowledge. A water resources engineer applies his expertise to conducting data analyses, modelling physical systems, and selecting practical and viable solutions. Water resources engineering experts rely on a combination of data, theoretical knowledge and operational experience to solve water resources problems. Defining how these are connected to produce expertise is important not only in expert system development, but for education and training in the future [49]. Developing expert systems becomes an extremely useful exercise for both knowledge engineer and expert. The knowledge engineer receives training in the domain, while the expert gains insight into his approach, which will improve his ability to transfer this knowledge in the future, even if an expert system is not created.

The relative amounts of the three problem solving assets, (i.e., data, theoretical knowledge and operational experience), seems to determine the ease with which heuristic rules are developed. In data analysis and modelling, more emphasis is often placed on data and theoretical knowledge, with a relatively small portion of experience and judgement. These more structured problems can make the burden of knowledge acquisition less cumbersome. On the other hand, at the final decision-making point, more skill and judgement are required. This is especially true when an expert is faced with political, environmental, and public concerns, and must still produce an economical design or plan, that meets some desired level of safety or reliability. Developing rules that adequately describe such a procedure may be close to impossible. It will, therefore, be impossible to remove the human expert from the decision-making process.

The impact of expert systems in water resources engineering will, thus, come not only at the level of designing intelligent systems, but also on the way engineering knowledge is transferred. These systems may become part of the training or education of future water resources engineers. Further work along the same lines as this thesis may lead to just such a future scenario.

REFERENCES

1. Alim, S., "A PROLOG-based Expert System for Encoding Seismic Design Provisions", Civil Engineering Systems, 4(1), 1987, p.39-44.

2. Allen, R.H., M.G. Boarnet, C.J. Culbert and R.T. Savely, "Using Hybrid Expert System Approaches for Engineering Applications", Engineering with Computers, 2(2), 1987, p.95-110.

3. Baffaut, C. and J.W. Delleur, "Expert Systems for Calibrating SWMM", Journal of Water Resources Planning and Management, 115(3), 1989, p.278-298.

4. Barritt-Flatt, P.E. and A.D. Cormie, "A Comprehensive Optimization Model for Hydro-Electric Reservoir Operations", paper presented at the ASCE Third Water Resources Operations and Management Workshop: Computerized Decision Support Systems for Water Managers, Fort Collins, Colorado, June, 1988.

5. Blockley, D.I., "Uncertainty Analysis in Expert Systems", Civil Engineering Systems, 4(1), p.3-6, 1987.

6. Bobrow, D.G., S. Mittal and M.J. Stefik, "Expert Systems: Perils and Promise", Communications of the ACM, 29(9), 1986, p.880-894.

7. Boose, J.H., Expertise Transfer for Expert System Design, Elsevier Science Publications, New York, 1986.

8. Bramer, M.A., "Expert Systems: The Vision and the Reality", in Research and Development in Expert Systems, M.A. Bramer (ed.), Cambridge University Press, New York, 1985, p.1-12.

9. Burgoyne, C.J. and S.H.R. Sham, "Application of Expert Systems to Prestressed Concrete Bridge Design", Civil Engineering Systems, 4(1), 1987, p.14-19.

10. Cohn, L.F., R.A. Harris and W. Bowlby, "Knowledge Acquisition for Domain Experts", Journal of Computing in Civil Engineering, 2(2), 1988, p.107-120.

11. Coombs, M. and J. Alty, "Expert Systems: An Alternative Paradigm", in Developments in Expert Systems, M.J. Coombs (ed.), Academic Press, Toronto, 1984, p.135-157.

12. Cormie, A.D. and P.E. Barritt-Flatt, "Hermes: A Decision Support System for Reservoir Operations at Manitoba Hydro", paper presented to Operation Planning Section, Canadian Electrical Association, Vancouver, B.C., March, 1987.

13. Cormie, A.D., P.E. Barritt-Flatt, R.W. Herzog, L. Kessler and R.

Welwood, EMMA: Energy Management and Maintenance Analysis, Manitoba Hydro.

14. Cupello, J.M. and D.J. Mishelevich, "Managing Prototype Knowledge/Expert System Projects", Communications of the ACM, 31(5), 1988, p.534-550.

15. Dotan, A. and D.C. Willer, "Instant Hydro Forecasting", Civil Engineering (ASCE), 56(8), 1986, p.60-63.

16. Dreyfus, H. and S., "Why Computers May Never Think Like People", Technology Review, 89(1), 1986, p.42-61.

17. Duda, R.O. and E.H. Shortliffe, "Expert Systems Research", Science, 220(4594), 1983, p.261-268.

18. Dym, C.L., "Expert Systems: New Approaches to Computer-aided Engineering", Engineering with Computers, 1(1), 1985, p.9-25.

19. Evans, M., Class Notes: 74.420 Expert Systems, Department of Computer Science, University of Manitoba, 1989.

20. Fayegh, D. and S.O. Russell, "An Expert System for Flood Estimation", in Expert Systems in Civil Engineering, ASCE Publication, New York, 1986, p.174-181.

21. Fedra, K., E. Weigkricht and L. Winkelbauer, "A Hybrid Approach to Information and Decision Support Systems: Hazardous Substances and Industrial Risk Management", Report RR-87-12, International Institute for Applied Systems Analysis, Laxenburg, Austria, 1987.

22. Fenves, S.J., M.L. Maher and D. Sriram, "Expert Systems: C.E. Potential", Civil Engineering (ASCE), 54(10), 1984, p.44-47.

23. Franck, B.M. and T. Krauthammer, "An Expert System for Field Inspection of Concrete Dams: Part 1, Engineering Knowledge", Engineering with Computers, 5(1), 1989, p.23-38.

24. Franck, B.M. and T. Krauthammer, "An Expert System for Field Inspection of Concrete Dams: Part 2, Artificial Intelligence Issues", Engineering with Computers, 5(2), 1989, p.119-131.

25. Haley, P. and C. Williams, "Expert System Development Requires Knowledge Engineering", Computer Design, 25(2), 1986, p.83-88.

26. Hart, A., Knowledge Acquisition for Expert Systems, McGraw-Hill Book Company, Toronto, 1986.

27. Hayes-Roth, F., "Knowledge-Based Expert Systems", Computer, 17(10), 1984, p.263-273.

28. Hjorth, P., "Future Prospects in Decision Support", Proceedings of the

International Conference on Decision Support Systems in Water Resources, Oslo, Norway, April, 1986.

29. Hogley, J.R. and A.R. Korncoff, "Artificial Intelligence in Engineering: A Revolutionary Change", in Applications of Artificial Intelligence in Engineering Problems, Volume II, D. Sriram and R. Adley (eds.), Computational Mechanics Publications, New York, 1986, p.1155-1160.

30. Jenkins, W.O. and P.W. Jowitt, "Expert Systems in River Basin Management", Civil Engineering Systems, 4(1), 1987, p.31-38.

31. Kahn, G., S. Nowlan and J. McDermott, "Strategies for Knowledge Acquisition", IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-7(5), 1985, p.511-522.

32. Kindler, J., "Water Resources Planning and Decision Making: Some Thoughts about the Future", Proceedings of the International Conference on Decision Support Systems in Water Resources, Oslo, Norway, April, 1986.

33. Kirby, H.R. and F.O. Montgomery, "Towards a Rule-based Approach for Traffic Signal Design", Civil Engineering Systems, 4(1), 1987, p.20-26.

34. Knight, B., "An Expert System for Upgrading Small Water Supplies", Civil Engineering Systems, 4(1), 1987, p.27-30.

35. Kolodner, J., "Towards an Understanding of the Role of Experience in the Evolution from Novice to Expert", in Developments in Expert Systems, M.J. Coombs (ed.), Academic Press, Toronto, 1984, p.95-116.

36. Kostem, C.N. and M.L. Maher (eds.), Expert Systems in Civil Engineering, ASCE Publication, New York, 1986.

37. Lenat, D.B., "The Nature of Heuristics", Artificial Intelligence, 19(2), 1982, p.189-249.

38. Lu, S.C-Y., "Knowledge Map: An Approach to Knowledge Acquisition in Developing Engineering Expert Systems", Engineering with Computers, 3(2), 1987, p.59-68.

39. Maher, M.L., Expert Systems for Civil Engineers: Technology and Application, ASCE Publication, New York, 1987.

40. National Research Council, Safety of Dams: Flood and Earthquake Criteria, National Academy Press, Washington, D.C., 1985.

41. Negoita, C.V., Expert Systems and Fuzzy Systems, The Benjamin/ Cummings Publishing Company Inc., Don Mills, Ontario, 1985.

42. Palmer, R.N. and K.J. Holmes, "Operational Guidance During Droughts: Expert Systems Approach", Journal of Water Resources Planning and Management, 114(6), 1988, p.647-666.

43. Palmer, R.N. and B.W. Mar, "Expert Systems Software for Civil Engineering Applications", Civil Engineering Systems, 5(4), 1988, p.170-180.

44. Palmer, R.N. and R.M. Tull, "Expert Systems for Drought Management Planning", Journal of Computing in Civil Engineering, 1(4), 1987, p.284-297.

45. Paperback Software International, VP-Expert, 1987.

46. Radian Corporation, RuleMaster 2: A Software Tool for Building Expert Systems, Reference Manual, 1987.

47. Reznicek, K., "An Application of Successive Linear Programming to the Optimization of an Interconnected Hydro Utility Operation", M.Sc. thesis, University of Manitoba, 1988.

48. Rolston, D.W., Principles of Artificial Intelligence and Expert Systems Development, McGraw-Hill Book Company, Toronto, 1988.

49. Russell, S.O.D., "The Design Process - Insights from Expert System Development", paper presented at the Fourth Canadian Seminar on Systems Theory for the Civil Engineer, Winnipeg, Manitoba, May, 1989.

50. Simonovic, S.P., "Short Course on: Expert Systems for Water Resources Engineers", Department of Civil Engineering, University of Manitoba, 1989.

51. Simonovic, S.P. and D.A. Savic, "Intelligent Decision Support and Reservoir Management and Operations", ASCE Journal of Computing in Civil Engineering, 3(3), in print, 1989.

52. Simonovic, S. and K. Barlishen, "Expert Systems in Water Resources Systems Analysis - Literature Review", Water Resources Research Report, No. 004, University of Manitoba, Department of Civil Engineering, August, 1987.

53. Starfield, A.M., K.L. Butala, M.M. England and K.A. Smith, "Mastering Engineering Concepts by Building an Expert System", Engineering Education, 74(2), 1983, p. 104-107.

54. Virard, M.A., "Transfer of Engineering Knowledge", Engineering Digest, 32(8), 1986, p.32-33.

55. Waterman, D.A., A Guide to Expert Systems, Addison-Wesley Publishing Company, Don Mills, Ontario, 1986.

APPENDIX A

Listing of Pre-processor to EMMA

```
/* FILE: input.ind */

MODULE: input
DECLARATIONS:
[
        CHILD: flow
        CHILD: pcoeff
        CHILD: stages
        LOCAL: string {k sys_title sys_name sys_help}
LOCAL: float return_value
LOCAL: integer {i tsteps num_inflows strips num_stations num_lakes}]

STATE: initial
[("*** ADVICE ON INPUT DATA FORM ***" -> sys_title;
"input.exe" -> sys_name;
"general.hlp" -> sys_help;
start;
s_print "***********************************************
* Some initial suggestions are going to be made *
* regarding the input data required by EMMA.    *
***********************************************", current)]

STATE: current
[IF (s_ask3 "Is the current description of the state of the system
included in the problem formulation?" "yes, no"
" "
"Are you starting the EMMA run from the present state of the
system? i.e. current reservoir levels, etc.") IS
"yes": (null, objective)
ELSE (s_adv3 "If the current state of the system is not included in the
analysis, it may lead to a solution that starts the system at a position
very different from the current. A large change in such things as lake
levels or releases are impossible to accomodate, invalidating the solution.
--> May want to include the current state in the analysis so
EMMA doesn't set an extremely different starting position."
" "
"Can include the current conditions in the
data for the first timestep of analysis.", objective)]

STATE: objective
[(s_print "If there is a particular objective you wish to include in the
analysis, it would be advised to try and accomplish it through the
objective function rather than the constraints.
EMMA may try to find a way around a constraint so an objective is not
met, so formulate it as part of the objective function.
eg. to achieve an objective of minimizing spill in a particular spillway,
perhaps because it hasn't been used for awhile and may not be able to
cope with the flow,
--> increase the costs associated with spilling
in that particular generating station input data.", export)]
```

```
STATE: export
[IF (s_ask "Are you currently in a drought condition?" "yes, no") IS
"yes": (s_adv3 "You may wish to hold onto water and not export energy,
--> assign a higher value to water in storage than the export value."
" "
"In the lake data, values for storage are assigned, and you can ensure
these values exceed the value set for exports, so EMMA will try and
avoid releasing water, due to the high cost.", worst)
ELSE (null, worst)]

STATE: worst
[IF (s_ask3 "Are you planning on considering a worst case scenario?"
"yes, no"
" "
"Are you going to analyze the system under low flows /high load?") IS
"yes": (s_adv3 "TO CREATE A WORST CASE SCENARIO:
- use inflow values at the 5% probability level
- increase expected loads by 10%, but make sure they are not
greater than the system capacity"
" "
"Inflow values are assumed to be log-normally distributed.
The load forecast should be checked against the combined
available energy production capability of the plants.", begin)
ELSE (s_adv3 "It may be helpful to analyze not only the most probable
case, but the worst case scenario. For example drought conditions and an
objective of conserving water could be analyzed for comparison.
Or, may include the worst case conditions in the data for the first time
step of the analysis to constrain the most probable case.
TO CREATE A WORST CASE SCENARIO:
- use inflow values at the 5% probability level
- increase expected loads by 10%, but < system capacity"
" "
"Inflows are assumed to be log-normally distributed.
The load forecast values should be checked against the combined available
energy production capability of the plants.", begin)]

STATE: begin
[(s_print "*****************************************************-
****
* Now, advice will be given regarding the values that have  *
* been set for some important variables. The comparison will *
* be with the normal historical ranges of these variables to *
* locate possible input data errors, before running EMMA.   *
*************************************************************", flows)]

STATE: flows
[IF (s_ask3 "Would you like to check the LOCAL INFLOWS?" "yes, no"
" "
"Inflows are assumed to be log-normally distributed. The input inflow
values will be checked against historical values to uncover possible data
input errors.") IS
"yes":(emma1 1 -> tsteps; emma99 1 -> num_inflows;
```

```
flow, production)
ELSE (null, production)]

STATE: production
[IF (s_ask3 "Would you like to check PRODUCTION COEFFICIENT values?" "yes, no"
" "
"Initial estimates for the production coefficients should
be based on the normal head of the generating station:
(Forebay_level - Tailwater Level) * efficiency / 11.820.
The assumed values will be checked against the ranges of
the storage curves. A good estimate will lead to quicker
convergence.") IS
"yes": (stations 1 -> num_stations;
pcoeff, initial_stages)
ELSE (null, initial_stages)]

STATE: initial_stages
[IF (s_ask3 "Would you like to check the STARTING RESERVOIR LEVELS?" "yes, no"
" "
"The initial reservoir stages will be checked against
the historical minimum and maximum values, to check
for possible input data errors") IS
"yes": (lakes 1 -> num_lakes;
stages, GOAL)
ELSE (finish, GOAL)]
```

```
/* FILE: flow.ind */

MODULE: input.flow
DECLARATIONS:
[INTENT: "checkIthe inflow valuesI"
LOCAL: float {x y z}
LOCAL: string {s_month d e}
LOCAL: integer {first_month index f inflow month}]

STATE: initialize
[(s_read "What is the first timestep (as an integer)?
i.e. JANUARY -> 1 ... DECEMBER -> 12" -> s_month;
s_to_i s_month -> month;
month - 1 -> month;
month -> first_month;
```

```
0 -> index;
0 -> inflow,
inflow_loop)]

STATE: inflow_loop
[IF (inflow < num_inflows) IS
"T": (null, test_month)
ELSE (null, GOAL)]

STATE: test_month
[IF (month == 12) IS
"T": (0 -> month, month_loop)
ELSE (null, month_loop)]

STATE: month_loop
[IF (index < tsteps) IS
"T": (flow_data inflow index -> x;
log x -> x;
hist_mean inflow month -> y;
log y -> y;
hist_dev inflow month -> z;
index + 1 -> index;
month + 1 -> month, compare99)
ELSE (first_month -> month;
0 -> index; inflow + 1 -> inflow, inflow_loop)]

STATE: compare99
[IF ( ( 2.33 * z + y < x) or (y - (2.33 * z) > x) ) IS
"T": (inflow + 1 -> f;
i_to_s f -> d;
i_to_s month -> e;
s_print ("WARNING!
The inflow value for the following local inflow and
month should be checked, as it is > 2 standard deviations
away from the historical mean, i.e. outside the 99
percentile probability range.") # ("

INFLOW (kcfs) = ") # d # ("
MONTH = ") # e, test_month)
ELSE (null, compare95)]

STATE: compare95
[IF ( ( 1.645 * z + y < x) or (y - (1.645 * z) > x) ) IS
"T": (inflow + 1 -> f;
i_to_s f -> d;
i_to_s month -> e;
s_print ("WARNING!
The following inflow value is outside of the
95 % probability range:") # ("

INFLOW (kcfs) = ") # d # ("
MONTH = ") # e, test_month)
```

```
ELSE (null, test_month)]
```

```
/* FILE: pcoeff.ind */

MODULE: input.pcoeff
DECLARATIONS:
[INTENT: "checkIthe values assigned to production coefficientl"
LOCAL: float {pc_min pc_max input_pc}
LOCAL: string {a b c d s_tstep s_strip}
LOCAL: integer {gsta_index month_index strip strips minmax}]

STATE: begin
[(s_print "** The values set for the starting production coefficients will
be compared to the historical ranges. If a value out of the range is
encountered, it will be pointed out. **", initial)]

STATE: initial
[(0 -> gsta_index;
0 -> month_index;
0 -> strip, station_loop)]

STATE: station_loop
[IF (gsta_index < num_stations) IS
"T": (0 -> minmax;
hist_pcoeff gsta_index minmax -> pc_min;
f_to_s pc_min -> b;
1 -> minmax;
hist_pcoeff gsta_index minmax -> pc_max;
f_to_s pc_max -> c, month_loop)
ELSE (null, GOAL)]

STATE: month_loop
[IF (month_index < tsteps) IS
"T": (load_strips month_index -> strips, strip_loop)
ELSE (0 -> month_index;
gsta_index + 1 -> gsta_index, station_loop)]

STATE: strip_loop
[IF (strip < strips) IS
"T": (pcs gsta_index strip month_index -> input_pc;
f_to_s input_pc -> a, minimum)
ELSE (0 -> strip;
month_index + 1 -> month_index, month_loop)]

STATE: minimum
[IF (pc_min > input_pc) IS
```

"T": (i_to_s gsta_index -> d;
i_to_s month_index -> s_tstep;
i_to_s strip -> s_strip;
s_print ("The value set for the production coefficient of the following generating station falls below the minimum advisable and should be adjusted:") # ("

GEN_STATION = ") # d # ("
TSTEP = ") # s_tstep # ("
STRIP = ") # s_strip # ("

REALISTIC MINIMUM = ") # b # ("
INPUT VALUE = ") # a;
strip + 1 -> strip, strip_loop)
ELSE (null, maximum)]

STATE: maximum
[IF (pc_max < input_pc) IS
"T": (i_to_s (gsta_index + 1) -> d;
i_to_s (month_index + 1) -> s_tstep;
i_to_s (strip + 1) -> s_strip;
s_print ("The value set for the production coefficient of the following generating station is greater than the maximum advisable and should be adjusted:") # ("

GEN_STATION = ") # d # ("
TSTEP = ") # s_tstep # ("
STRIP = ") # s_strip # ("

REALISTIC MAXIMUM = ") # c # ("
INPUT VALUE = ") # a;
strip + 1 -> strip, strip_loop)
ELSE (strip + 1 -> strip, strip_loop)]

/* FILE: stages.ind */

MODULE: input.stages
DECLARATIONS:
[INTENT: "checklthe values set for starting reservoir levelsl"
LOCAL: float {init_stage hist_min hist_max}
LOCAL: string {a b c d}
LOCAL: integer {lake_index minmax}]

STATE: begin
[(s_print "** The values set for the starting reservoir levels will be compared to the historical ranges. If a value

```
out of the range is encountered, it will be pointed out. **", initialize)]

STATE: initialize
[(0 -> lake_index, lake_loop)]

STATE: lake_loop
[IF (lake_index < num_lakes) IS
"T": (0 -> minmax;
levels lake_index minmax -> hist_min;
f_to_s hist_min -> b;
1 -> minmax;
levels lake_index minmax -> hist_max;
f_to_s hist_max -> c;
input_stage lake_index -> init_stage;
f_to_s init_stage -> a;
lake_index + 1 -> lake_index, minimum)
ELSE (null, GOAL)]

STATE: minimum
[IF (hist_min > init_stage) IS
"T": (i_to_s lake_index -> d;
s_print ("The value set for the starting level of the following lake falls
below the historical minimum:") # ("

LAKE = ") # d # ("
HISTORICAL MINIMUM (ft) = ") # b # ("
INPUT VALUE (ft) = ") # a, lake_loop)
ELSE (null, maximum)]

STATE: maximum
[IF (hist_max < init_stage) IS
"T": (i_to_s lake_index -> d;
s_print ("The value set for the starting level of the following lake is
greater than the historical maximum:") # ("

LAKE = ") # d # ("
HISTORICAL MAXIMUM (ft) = ") # c # ("
INPUT VALUE (ft) = ") # a, lake_loop)
ELSE (null, lake_loop)]
```

```
/* EXTERNAL C FUNCTIONS */


/* FILE: emma1.c */

#include <stdio.h>
#include "/sw/emma_src/emma.ins.c"

long emma1(i)

long i;
{
 start_emma();
 return(em.NUM_TSTEPS);
}




/* FILE: emma99.c */

#include <stdio.h>
#include "/sw/emma_src/emma.ins.c"

long emma99(i)

long i;
{
 return(em.NUM_LOC_INFLOWS);
}




/* FILE: stations.c */
```

```
#include <stdio.h>
#include "/sw/emma_src/emma.ins.c"

long stations(i)

long i;
{
 return(em.NUM_GSTATIONS);
}
```

```
/* FILE: lakes.c */

#include <stdio.h>
#include "/sw/emma_src/emma.ins.c"

long lakes(i)

long i;
{
 return(em.NUM_LAKES);
}
```

```
/* FILE: flow_data.c */

#include <stdio.h>
#include "/sw/emma_src/emma.ins.c"

#include "RMccode.h"
#include "RMextern.h"


double flow_data(inflow, index)


long inflow, index;
{
 return(em.LOC_INFLOW[inflow].FLOW[index]);
}
```

```
/* FILE: hist_mean.c */

#include <stdio.h>
#include "RMccode.h"
#include "RMextern.h"

float
hist_mean(inflow, month)
long inflow, month;
{
static char *name[20] ={
                "Laurie River",
                "Russell River",
                "Wilmot Creek",
                "Tippieskaw Creek"
                };
static float means[4][12] ={
                 {0.25, 0.22, 0.35, 0.45, 0.55, 0.45, 0.45, 0.43, 0.40, 0.30, 0.25,
0.25},
                 {0.38, 0.42, 0.50, 0.53, 0.62, 0.57, 0.52, 0.54, 0.55, 0.50, 0.45,
0.40},
                 {0.06, 0.08, 0.09, 0.10, 0.11, 0.15, 0.13, 0.11, 0.100, 0.090,
0.075, 0.065},
                 {0.06, 0.07, 0.09, 0.11, 0.125, 0.13, 0.145, 0.150, 0.100, 0.09,
0.075, 0.065}

};
                 return(means[inflow][month]);
}



/* FILE: hist_dev.c */

#include <stdio.h>
#include "RMccode.h"
#include "RMextern.h"

float
hist_dev(inflow, month)
long inflow, month;
 {
static char *name[20] ={
                "Laurie River",
                "Russell River",
                "Wilmot Creek",
                "Tippieskaw Creek"
                };
static float deviation[4][12] ={
```

```
                {0.05, 0.04, 0.035, 0.045, 0.055, 0.045, 0.045, 0.043, 0.040,
0.030, 0.025, 0.025},
                {0.038, 0.042, 0.050, 0.053, 0.062, 0.057, 0.052, 0.054, 0.055,
0.050, 0.045, 0.040},
                {0.01, 0.02, 0.02, 0.01, 0.011, 0.015, 0.013, 0.011, 0.010,
0.0090, 0.0075, 0.0065},
                {0.006, 0.007, 0.009, 0.011, 0.0125, 0.013, 0.0145, 0.0150, 0.010,
0.009, 0.0075, 0.0065}

};
                return(deviation[inflow][month]);
}
```

```
/* FILE: hist_pcoeff.c */

#include <stdio.h>
#include "RMccode.h"
#include "RMextern.h"

float
hist_pcoeff(gsta_index, minmax)
long gsta_index, minmax;
{
static char *name[20] ={
                "Laurie #2",
                "Laurie #1"
                };
static float coeff[2][2] ={
                {2.91, 4.64},
                {4.09, 4.67}

};
                return(coeff[gsta_index][minmax]);
}
```

```
/* FILE: load_strips.c */

#include <stdio.h>
#include "/sw/emma_src/emma.ins.c"

long load_strips(month_index)
```

```
long month_index;
{
 return(em.LDC[month_index].NUM_STRIPS);
}



/* FILE: pcs.c */

#include <stdio.h>
#include "/sw/emma_src/emma.ins.c"

#include "RMccode.h"
#include "RMextern.h"

double pcs(gsta_index, strip, month_index)

long gsta_index, strip, month_index;
{
 return(em.GSTATION[gsta_index].PROD_COEFF[strip][month_index]);
}



/* FILE: levels.c */

#include <stdio.h>
#include "RMccode.h"
#include "RMextern.h"

float
levels(lake, minmax)
long lake, minmax;
{
static char *name[20] ={
                "Eager Lake",
                "Russell Lake",
                "McGavock Lake",
                "Laurie #2 Forebay",
                "Laurie #1 Forebay"
                };
static float reservoir[5][2] ={
                 {101.0, 114.0},
                 {102.0, 114.0},
                 {108.0, 118.0},
                 {208.0, 213.0},
                 {156.0, 156.0}
```

```
};
                return(reservoir[lake][minmax]);
}
```

APPENDIX B

Listing of Post-processor to EMMA

```
/*  FEASIBLE CASE  */

/*  FILE: output.ind  */

MODULE: output
DECLARATIONS:
[
        CHILD: release
        CHILD: changes
        CHILD: levels
        CHILD: level_changes
        LOCAL: string {s_coeff sys_title sys_name sys_help}
LOCAL: float return_value
LOCAL: integer {first_month coeff_files max_iterations i}
LOCAL: integer {tsteps num_lakes num_gstations num_control_flows}]

STATE: title
[("***  ADVICE  ON  THE  PRACTICALITY  OF  THE  EMMA  OUTPUT
RESULTS ***" -> sys_title;
"input.exe" -> sys_name;
"general.hlp" -> sys_help;
start;
s_print "************************************************
* Some initial suggestions are going to be made *
* regarding the output generated by EMMA.      *
************************************************", convergence)]

STATE: convergence
[(emma_in 1 -> tsteps;
s_read3  "How  many  pcoeff  files  were  generated  as  output  files  for  this
particular run?"
" "
"The files will be located in the solutions directory of the
given  input  filename.  The  pcoeff  files  are  numbered  and  listed."  ->
s_coeff;
s_to_i s_coeff -> coeff_files;
iterations 1 -> max_iterations, convergence_advice)]

STATE: convergence_advice
[IF (coeff_files == max_iterations) IS
"T": (s_adv3 "The production coefficients have not converged yet.
Rerun the problem as it now stands."
" "
"The system has reached the maximum number of
allowed coefficient iterations. The problem
should be run again as it now stands until this
convergence is accomplished. Convergence may be
slow if initial estimates for these values were
not good. If EMMA is having a very difficult time
with convergence, the system may not be segmetized
properly.", errors)
ELSE (s_adv "The production coefficients have converged!!", errors)]
```

```
STATE: errors
[IF (s_ask3 "Is there an error listed at the top of the lp_soln file?
i.e. min/max flow intercept <> 0 for a control structure" "yes, no"
" "
"This error should be automatically noticed by EMMA, but
sometimes it is not. The intercepts for all but the first
segment of storage should be zero as the flow-storage curve
is non-cumulative") IS
"yes": (s_adv3 "This error must be corrected and the problem rerun with
a previously feasible basis if it exists. The
control structure involved is indicated."
" "
"In the control structure data, values for flow intercepts
are assigned, and you must ensure that they are zero for
all but the first segment because flow-storage curves are
non-cumulative.", solutions)
ELSE (null, solutions)]

STATE: solutions
[(s_adv3 "Check to make sure that the primal and dual solutions for the
objective function value are the same. These values are ZX and ZY in the
lp_soln file. If they are not the same rerun the problem with a previous
feasible basis if one exists."
" "
"The objective function values derived from the primal
and dual problems must be the same for the solution to
be feasible and optimal.", continue)]

STATE: continue
[IF (s_ask "Continue the analysis?" "yes, no") IS
"yes": (null, begin)
ELSE (finish, GOAL)]

STATE: begin
[(s_print "**********************************************************_
****
* Now, advice will be given regarding the values that have  *
* been set for some important output variables. Some of the  *
* comparisons be with the normal historical ranges for these *
* variables to determine if the results are practical. Other *
* advice will be given based on additional environmental and *
* social factors that must be considered .                   *
**********************************************************",
assignments)]

STATE: assignments
[(c_flows 1 -> num_control_flows;
gstas 1 -> num_gstations;
lakes 1 -> num_lakes;
release;
changes;
levels;
```

```
level_changes;
finish, GOAL)]
```

```
/*  FILE: release.ind  */

MODULE: output.release
DECLARATIONS:
[INTENT: "checkIthe release values!"
LOCAL: float {x y z}
LOCAL: string {s_month b c d e}
LOCAL: integer {index f gstation month}
LOCAL: float {ph_flow spillage sum_flow}]

STATE: initialize
[(s_read "What is the first timestep (as an integer)?
i.e. JANUARY -> 1 ... DECEMBER -> 12" -> s_month;
s_to_i s_month -> month;
month - 1 -> month;
month -> first_month;
0 -> index;
0 -> gstation;
s_print "**************************************
** CHECK RELEASE VALUES SET BY EMMA **
**************************************",
gsta_loop)]

STATE: gsta_loop
[IF (gstation < num_gstations) IS
"T": (null, test_month)
ELSE (null, GOAL)]

STATE: test_month
[IF (month == 12) IS
"T": (0 -> month, month_loop)
ELSE (null, month_loop)]

STATE: month_loop
[IF (index < tsteps) IS
"T": (total_flow gstation index -> sum_flow;
spill_flow gstation index -> spillage;
sum_flow - spillage -> ph_flow;
max_release gstation month -> y;
min_release gstation month -> z;
index + 1 -> index;
```

```
month + 1 -> month, compare_max)
ELSE (first_month -> month;
0 -> index; gstation + 1 -> gstation, gsta_loop)]

STATE: compare_max
[IF (ph_flow > y) IS
"T": (gstation + 1 -> f;
i_to_s f -> d;
i_to_s month -> e;
f_to_s ph_flow -> b;
f_to_s y -> c;
s_print ("WARNING!
The release scheduled for a generating station is
larger than the historical maximum release. The
generating station has not been run at this level
before. You may want to constrain the maximum flow
through the powerhouse to this historical maximum.") # ("

GSTATION = ") # d # ("
MONTH = ") # e # ("
SET RELEASE (kcfs) = ") # b # ("
HISTORICAL MAXIMUM (kcfs) = ") # c, test_month)
ELSE (null, compare_min)]

STATE: compare_min
[IF (ph_flow < z) IS
"T": (gstation + 1 -> f;
i_to_s f -> d;
i_to_s month -> e;
f_to_s ph_flow -> b;
f_to_s z -> c;
s_print ("WARNING!
The release scheduled for a generating station falls
below the historical minimum release. The generating
station has not been run at this low level before.") # ("

GSTATION = ") # d # ("
MONTH = ") # e # ("
SET RELEASE (kcfs)  = ") # b # ("
HISTORICAL MINIMUM (kcfs) = ") # c, test_month)
ELSE (null, test_month)]



/*  FILE: changes.ind  */

MODULE: output.changes
DECLARATIONS:
[INTENT: "checkIthe values of release changesI"
LOCAL: float {delta y z}
LOCAL: string {b c d e}
```

```
LOCAL: integer {index control_structure counter f gstation month}
LOCAL: float {release1 release2 spillage1 spillage2 sum_flow1 sum_flow2}]

STATE: initialize
[(s_print "************************************
** CHECK VALUES OF RELEASE CHANGES **
************************************";
first_month -> month;
0 -> index;
1 -> counter;
0 -> gstation,
gsta_loop)]

STATE: gsta_loop
[IF (gstation < num_gstations) IS
"T": (null, gtest_month)
ELSE (null, control_init)]

STATE: gtest_month
[IF (month == 12) IS
"T": (0 -> month, gmonth_loop)
ELSE (null, gmonth_loop)]

STATE: gmonth_loop
[IF (counter < tsteps) IS
"T": (total_flow gstation index -> sum_flow1;
total_flow gstation counter -> sum_flow2;
spill_flow gstation index -> spillage1;
spill_flow gstation counter -> spillage2;
sum_flow1 - spillage1 -> release1;
sum_flow2 - spillage2 -> release2;
sqrt ((-1.0) * (release1 - release2) * (release2 - release1)) -> delta;
max_change gstation month -> y;
index + 1 -> index;
counter + 1 -> counter;
month + 1 -> month, gcompare_max)
ELSE (first_month -> month;
0 -> index;
1 -> counter;
gstation + 1 -> gstation, gsta_loop)]

STATE: gcompare_max
[IF (delta > y) IS
"T": (gstation + 1 -> f;
i_to_s f -> d;
i_to_s month -> e;
f_to_s delta -> b;
f_to_s y -> c;
s_print ("WARNING! -> PH RELEASE CHANGE

A release change required between two timesteps is
larger than the historical maximum release change.
```

This change cannot likely be adequately handled by
the plant. It must be done gradually, whereas,
EMMA assumes instantaneous changes are possible.

As well, such a large change may upset fish and human
populations downstream.

--> It may be necessary to constrain this change.
This may be done by setting a low value for the
maximum delta storage for the upstream reservoir.
Remember, a zero means no limit !!

```
GSTATION = '') # d # ('
STARTING MONTH = '') # e # ('
RELEASE CHANGE TO NEXT TIMESTEP (kcfs) = '') # b # ('
HISTORICAL MAXIMUM (kcfs) = '') # c, gtest_month)
ELSE (null, gtest_month)]

STATE: control_init
[(0 -> index;
0 -> control_structure;
1 -> counter, control_loop)]

STATE: control_loop
[IF (control_structure < num_control_flows) IS
"T": (null, ctest_month)
ELSE (null, GOAL)]

STATE: ctest_month
[IF (month == 12) IS
"T": (0 -> month, cmonth_loop)
ELSE (null, cmonth_loop)]

STATE: cmonth_loop
[IF ((index < tsteps) and (counter < tsteps)) IS
"T": (
control_flows control_structure index -> release1;
control_flows control_structure counter -> release2;
sqrt ((-1.0) * (release1 - release2) * (release2 - release1)) -> delta;
max_control_change control_structure month -> y;
index + 1 -> index;
counter + 1 -> counter;
month + 1 -> month, compare_max)
ELSE (first_month -> month;
0 -> index;
1 -> counter;
control_structure + 1 -> control_structure, control_loop)]

STATE: compare_max
[IF (delta > y) IS
"T": (control_structure + 1 -> f;
i_to_s f -> d;
```

```
i_to_s month -> e;
f_to_s delta -> b;
f_to_s y -> c;
s_print ("WARNING! -> CONTROL STRUCTURE RELEASE CHANGE

A release change required between two timesteps is
larger than the historical maximum release change.
This change may not be practical, as it must be
done gradually, whereas EMMA assumes immediate
responses are possible.

Such a large change may upset fish and human
populations downstream.

--> It may be necessary to constrain this change.
This may be done by setting a low value for the
maximum delta storage for the upstream reservoir.
Remember, a zero means no limit !!

CONTROL STRUCTURE = ") # d # ("
STARTING MONTH = ") # e # ("
RELEASE CHANGE TO NEXT TIMESTEP (kcfs) = ") # b # ("
HISTORICAL MAXIMUM (kcfs) = ") # c, ctest_month)
ELSE (null, ctest_month)]
```

```
/*  FILE: levels.ind  */

MODULE: output.levels
DECLARATIONS:
[INTENT: "check|the reservoir stages|"
LOCAL: float {stage x y z}
LOCAL: string {s_month b c d e}
LOCAL: integer {index f lake month}]

STATE: initialize
[(first_month -> month;
0 -> index;
0 -> lake;
s_print "****************************************
** CHECK RESERVOIR LEVELS SET BY EMMA **
****************************************",
lake_loop)]

STATE: lake_loop
```

```
[IF (lake < num_lakes) IS
"T": (null, test_month)
ELSE (null, GOAL)]

STATE: test_month
[IF (month == 12) IS
"T": (0 -> month, month_loop)
ELSE (null, month_loop)]

STATE: month_loop
[IF (index < (tsteps + 1)) IS
"T": (lake_stages lake index -> stage;
stage_limits lake 0 -> y;
stage_limits lake 1 -> z;
index + 1 -> index;
month + 1 -> month, compare_max)
ELSE (first_month -> month;
0 -> index; lake + 1 -> lake, lake_loop)]

STATE: compare_max
[IF (stage > z) IS
"T": (lake + 1 -> f;
i_to_s f -> d;
i_to_s month -> e;
f_to_s stage -> b;
f_to_s z -> c;
s_print ("WARNING!
The stage set for a lake has exceeded the historical
maximum stage. The size of the exceedance may cause
problems for communities along the lake.

-> The maximum level specified in the input data for
this lake may be set closer to this historical value.

LAKE = ") # d # ("
MONTH = ") # e # ("
INITIAL STAGE (ft) = ") # b # ("
HISTORICAL MAXIMUM (ft) = ") # c, test_month)
ELSE (null, compare_min)]

STATE: compare_min
[IF (stage < y) IS
"T": (lake + 1 -> f;
i_to_s f -> d;
i_to_s month -> e;
f_to_s stage -> b;
f_to_s y -> c;
s_print ("WARNING!
The level of this lake falls below the historical
minimum and may cause problems for lake-front
dwellers.
```

```
It would be advised to constrain the allowable
minimum elevation for this lake closer to this
minimum value, especially if the difference in
levels is large.

LAKE = ") # d # ("
MONTH = ") # e # ("
INITIAL STAGE (ft) = ") # b # ("
HISTORICAL MINIMUM (ft) = ") # c, test_month)
ELSE (null, test_month)]



/*  FILE: level_changes.ind  */

MODULE: output.level_changes
DECLARATIONS:
[INTENT: "checkIthe values of changes in lake stagesl"
LOCAL: float {delta y z}
LOCAL: string {b c d e}
LOCAL: integer {index lake counter f month}
LOCAL: float {stage1 stage2}]

STATE: initialize
[(s_print "****************************************
** CHECK VALUES OF LAKE STAGE CHANGES **
****************************************";
first_month -> month;
0 -> index;
1 -> counter;
0 -> lake,
lake_loop)]

STATE: lake_loop
[IF (lake < num_lakes) IS
"T": (null, test_month)
ELSE (null, GOAL)]

STATE: test_month
[IF (month == 12) IS
"T": (0 -> month, month_loop)
ELSE (null, month_loop)]

STATE: month_loop
[IF (counter < (tsteps + 1)) IS
"T": (lake_stages lake index -> stage1;
```

```
lake_stages lake counter -> stage2;
sqrt ((-1.0) * (stage1 - stage2) * (stage2 - stage1)) -> delta;
max_lake_change lake month -> y;
index + 1 -> index;
counter + 1 -> counter;
month + 1 -> month, compare_max)
ELSE (first_month -> month;
0 -> index;
1 -> counter;
lake + 1 -> lake, lake_loop)]

STATE: compare_max
[IF (delta > y) IS
"T": (lake + 1 -> f;
i_to_s f -> d;
i_to_s month -> e;
f_to_s delta -> b;
f_to_s y -> c;
s_print ("WARNING! -> LAKE STAGE CHANGE

A change in lake levels required between two timesteps
is larger than the historical maximum stage change.
Such a large change may upset humans living downstream.
They must be warned in advance if lake levels are going
to fluctuate greatly.

As well, in winter months, large stage changes may cause
damage to dams due to the resulting ice breakup. The
suggested maximum change reflects this fact.

--> It may be necessary to constrain this change.
This may be done by setting a low value for the
maximum delta storage for this lake.
Remember, a zero means no limit !!

LAKE = ") # d # ("
STARTING MONTH = ") # e # ("
STAGE CHANGE TO NEXT TIMESTEP (ft) = ") # b # ("
MAXIMUM CHANGE (ft) = ") # c, test_month)
ELSE (null, test_month)]
```

```
/* EXTERNAL C FUNCTIONS */


/* FILE: emma_in.c */

#include <stdio.h>
#include "/sw/emma_src/emma.ins.c"

long emma_in(i)

long i;
{
 start_emma();
 return(em.NUM_TSTEPS);
}




/* FILE: iterations.c */

#include <stdio.h>
#include "/sw/emma_src/emma.ins.c"

long iterations(i)

long i;
{
 return(em.MAX_PROD_COEFF_ITERS);
}




/* FILE: c_flows.c */

#include <stdio.h>
#include "/sw/emma_src/emma.ins.c"

long c_flows(i)

long i;
{
 return(em.NUM_CONTRL_FLOWS);
}
```

```
/*  FILE: gstas.c  */

#include <stdio.h>
#include "/sw/emma_src/emma.ins.c"

long gstas(i)

long i;
{
 return(em.NUM_GSTATIONS);
}
```

```
/*  FILE: lakes.c  */

#include <stdio.h>
#include "/sw/emma_src/emma.ins.c"

long lakes(i)

long i;
{
 return(em.NUM_LAKES);
}
```

```
/*  FILE: total_flow.c  */

#include <stdio.h>
#include "RMccode.h"
#include "RMextern.h"

float
total_flow(gstation, index)
long gstation, index;
{
static char *name[20] ={
                "Laurie #1",
                "Laurie #2"
                };
static float total[2][12] ={
                 {1.011, 0.844, 0.785, 0.911, 1.146},
                 {1.011, 0.844, 0.785, 0.911, 1.146}
};
                 return(total[gstation][index]);
}
```

```
/*  FILE: spill_flow.c  */

#include <stdio.h>
#include "RMccode.h"
#include "RMextern.h"

float
spill_flow(gstation, index)
long gstation, index;
{
static char *name[20] ={
                    "Laurie #1",
                    "Laurie #2"
                    };
static float spill[2][12] ={
                     {0.0, 0.0, 0.0, 0.0, 0.012},
                     {0.0, 0.0, 0.0, 0.0, 0.0}
};
                     return(spill[gstation][index]);
}




/*  FILE: max_release.c  */

#include <stdio.h>
#include "RMccode.h"
#include "RMextern.h"

float
max_release(gstation, month)
long gstation, month;
{
static char *name[20] ={
                    "Laurie #1",
                    "Laurie #2"
                    };
static float hist_max[2][12] ={
                     {1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},
                     {1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1, 1.1}
};
                     return(hist_max[gstation][month]);
}
```

```
/*  FILE: min_release.c  */

#include <stdio.h>
#include "RMccode.h"
#include "RMextern.h"

float
min_release(gstation, month)
long gstation, month;
{
static char *name[20] ={
                "Laurie #1",
                "Laurie #2"
                };
static float hist_min[2][12] ={
                 {0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8},
                 {0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8, 0.8}
};
                 return(hist_min[gstation][month]);
}



/*  FILE: max_change.c  */

#include <stdio.h>
#include "RMccode.h"
#include "RMextern.h"

float
max_change(gstation, month)
long gstation, month;
{
static char *name[20] ={
                "Laurie #1",
                "Laurie #2"
                };
static float hist_max[2][12] ={
                 {0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2},
                 {0.15,  0.15,  0.15,  0.15,  0.15,  0.15,  0.15,  0.15,  0.15,  0.15,
0.15}
};
                 return(hist_max[gstation][month]);
}
```

```
/*  FILE: control_flows.c  */

#include <stdio.h>
#include "RMccode.h"
#include "RMextern.h"

float
control_flows(control_structure, index)
long control_structure, index;
{
static char *name[20] ={
                    "Eager Dam",
                    "Russell Dam"
                    };
static float flow[2][12] ={
                     {0.966, 0.269, 0.524, 0.200, 0.240},
                     {0.711, 0.200, 0.200, 0.381, 0.726}
};
                     return(flow[control_structure][index]);
}



/*  FILE: max_control_change.c  */

#include <stdio.h>
#include "RMccode.h"
#include "RMextern.h"

float
max_control_change(control_structure, month)
long control_structure, month;
{
static char *name[20] ={
                   "Eager Dam",
                   "Russell Dam"
                   };
static float hist_max[2][12] ={
                    {0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5},
                    {0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.3}
};
                    return(hist_max[control_structure][month]);
}
```

```
/*  FILE: lake_stages.c  */

#include <stdio.h>
#include "RMccode.h"
#include "RMextern.h"


float lake_stages(lake, index)


long lake, index;
{
static char *name[20] ={
                    "Eager Lake",
                    "Russell Lake",
                    "McGavock Lake",
                    "Laurie 2 Forebay",
                    "Laurie 1 Forebay"
                    };
static float elevation[5][13] ={
                     {110.50, 109.33, 109.40, 108.83, 108.94, 1080.96},
                     {108.00, 107.63, 108.15, 108.56, 108.59, 108.00},
                     {108.00, 115.33, 113.25, 114.16, 112.04, 111.41},
                     {213.00, 214.00, 214.00, 214.00, 213.00, 214.00},
                     {156.00, 156.00, 156.00, 156.00, 156.00, 156.00}
};
                     return(elevation[lake][index]);
}




/*  FILE: stage_limits.c  */

#include <stdio.h>
#include "RMccode.h"
#include "RMextern.h"

float
stage_limits(lake, i)
long lake, i;
{
static char *name[20] ={
                    "Eager Lake",
                    "Russell Lake",
                    "McGavock Lake",
                    "Laurie #2 Forebay",
                    "Laurie #1 Forebay"
                    };
static float hist_level[5][2] ={
                     {108.90, 110.0},
```

```
                    {105.00, 109.00},
                    {108.00, 114.00},
                    {210.00, 215.00},
                    {150.00, 160.00}
};
                    return(hist_level[lake][i]);
}



/*  FILE: max_lake_change.c  */

#include <stdio.h>
#include "RMccode.h"
#include "RMextern.h"

float
max_lake_change(lake, month)
long lake, month;
{
static char *name[20] ={
                    "Eager Lake",
                    "Russell Lake",
                    "McGavock Lake",
                    "Laurie 2 Forebay",
                    "Laurie 1 Forebay"
                    };
static float delta_stage[5][12] ={
                    {3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0},
                    {2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0},
                    {3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0, 3.0},
                    {0.9, 0.9, 1.0, 2.0, 2.0, 2.0, 2.0, 2.0, 0.9, 0.9, 0.9, 0.1},
                    {1.0, 1.0, 1.0, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0, 1.0, 1.0, 1.0}
};
                    return(delta_stage[lake][month]);
}
```

```
/*  INFEASIBLE CASE  */

/*  FILE: infeasible.ind  */

MODULE: infeasible

DECLARATIONS:
[INTENT: "adviselon infeasibilityl"
      CHILD: prelim
    LOCAL: string {sys_name sys_title sys_help}
 LOCAL: string {error peak num_constraints feasible}
 LOCAL:  string  {supply  flow_cont  tie_load  fraction  tie_min  genflow
conflow waste}
 LOCAL:  string  {strip_group  maint_con  crew  reqd_maint  space  avail_cap
crew_avail}
 LOCAL: string {run_river hydro_pro hshape gshape}]

STATE: begin
[("infeasible.exe" -> sys_name;
"Infeasibility Advisor" -> sys_title;
"general.hlp" -> sys_help;
start, initial)]

STATE: initial
[(s_print      "**********************************************************-
**********
*  This  expert  system  gives  advice  to  a  user  who  is  faced  with  an
infeasible *
*  solution  to  their  energy  management  problem.
      *
*  The  analysis  will  begin  with  some  initial  checks  on  the  solution
obtained. *
**********************************************************************-
**********";
prelim, final_advice)]

STATE: final_advice
[(s_print "The cause of an infeasibility is usually bad or incorrect data,
especially when the analysis began with a feasible system. If the data
appears correct, the problem cannot be solved as it stands. It must be
given more capacity somewhere in the system. Due to the infeasible
state of the problem, the output reports contain alot of garbage and are
of little help.

--> A suggestion would be to set up a dummy power plant that has large
capacity but high cost, so EMMA will try not to use it but will if it has
to in order to be feasible. Then the output reports can be examined
closely to see how the system is behaving.

THE ANALYSIS HAS BEEN COMPLETED!!!";finish, GOAL)]
```

```
/* FILE: prelim.ind */

MODULE: infeasible.prelim
DECLARATIONS:
[INTENT: "adviselon important initial output checks|"
        CHILD: constraints
        ]

STATE: initial
[(s_ask3 "Is there an error message listed at the start of the log file?
i.e. min/max flow intercept <> 0"
"yes, no"
"establish whether|an obvious error was made|"
"This error is one EMMA initially checks and must be corrected. The
error message will appear as the first line in the log output file."
-> error,next)]

STATE: next
[IF error IS
"yes": (s_adv3 "Correct the error and rerun the problem."
"correct flow intercept data and rerun EMMA"
"The location of the error is indicated in the error message.", GOAL)
ELSE (s_adv "The analysis will be continued.",peaking)]

STATE: peaking
[(s_ask3 "Is a peaking constraint indicated in the lp_soln file?" "yes, no"
"establish whether|the peaking bug was encountered|"
"The constraints involved in the infeasibility are underlined in the lp_soln
file|"
-> peak, peak_adv)]

STATE: peak_adv
[IF peak IS
"yes": (s_adv3 "If a maximum change in storage has not been set for the
lake indicated by the constraint (i.e, equal to zero) in the lake input data
EMMA has hit a bug.

--> Rerun the problem."
"rerun problem due to bug"
"The peaking constraint is of the form PEAKING(strip,lake,tstep).
It should only be created if a limit on the maximum change in lake
storage has been set. There is a bug in the EMMA program that causes it
to create this constraint when no limit has been specified.", GOAL)
ELSE (s_adv "The analysis will be continued.", changes)]

STATE: changes
[(s_ask3 "Did you begin with a feasible system and caused an
infeasibility by changing some input data?" "yes, no"
"know whether|it was initially a feasible system|"
"Was the previous run feasible? If so one of the changes made is the
cause of the infeasibility."
-> feasible, change_adv)]
```

```
STATE: change_adv
[IF feasible IS
"yes": (s_adv "If only a few changes have been made to the feasible solution you may want to make the changes one at a time to see which one is causing the infeasibility.", final)
ELSE (null, final)]

STATE: final
[IF (s_ask "Continue the analysis?" "yes, no") IS
"yes": (s_ask "Are relatively few constraints indicated (i.e. underlined) in the lp_soln file, say around 3?" "yes, no" -> num_constraints, relax)
ELSE (null, GOAL)]

STATE: relax
[IF num_constraints IS
"yes": (s_adv3 "--> You may want to relax the indicated constraints one at a time to see which one is causing the problem and check the input data involved in this constraint."
" "
"The EMMA manual will list the input data involved in a particular constraint.", end)
ELSE (null, end)]

STATE: end
[IF (s_ask "Continue the analysis?" "yes, no") IS
"yes": (constraints, GOAL)
ELSE (null, GOAL)]
```

```
/* FILE: constraints.ind */

MODULE: infeasible.prelim.constraints

DECLARATIONS:
[INTENT: "asklthe user about indicated constraintsl"
      CHILD: analysis]

STATE: indicated
[(s_print "*********************************************************
* In the lp_soln file, EMMA underlines variables and *
* constraints which are direcly involved in the      *
* infeasibility. You will be asked about constraints *
* that have been pointed to, so advice can be given  *
* regarding possible problem areas in the input data *
* file that may be responsible for the infeasibility. *
* Many problems are simply the result of data entry  *
```

```
* errors.                                        *
**************************************************";
s_ask "Was the SUPP_AND_DEMAND constraint indicated (i.e.
underlined) in the lp_soln file?" "yes, no" -> supply;
s_ask "FLOW_CONTIN ?" "yes, no" -> flow_cont;
s_ask "TLINE_LOAD ?" "yes, no" -> tie_load;
s_ask "IM_FRAC_LIM ?" "yes, no" -> fraction;
s_ask "TLINE_MIN ?" "yes, no" -> tie_min;
s_ask "GENFLOW_BND ?" "yes, no" -> genflow;
s_ask "CONFLOW_BND ?" "yes, no" -> conflow;
s_ask "WASTE_HEAT ?" "yes, no" -> waste;
s_ask "STRIP_GROUP ?" "yes, no" -> strip_group;
s_ask "Were some maintenance constraints indicated ?" "yes, no" ->
maint_con, continuity)]

STATE: continuity
[IF flow_cont IS
"yes": (null, maintenance)
ELSE (s_ask "RUN_OF_RIVER ?" "yes, no" -> run_river;
s_ask "HYDRO_PROD ?" "yes, no" -> hydro_pro;
s_ask "HYDRO_SHAPE ?" "yes, no" -> hshape;
s_ask "GENSRC_SHAPE ?" "yes, no" -> gshape, maintenance)]

STATE: maintenance
[IF maint_con IS
"yes": (s_ask "CREW_SCHEDULE ?" "yes, no" -> crew;
s_ask "HYDRO_REQD_MAINT or GENSRC_REQD_MAINT ?" "yes, no" ->
reqd_maint;
s_ask "MAINT_SPACE ?" "yes, no" -> space;
s_ask "HYDRO_AVAIL_CAP or GENSRC_AVAIL_CAP ?" "yes, no" ->
avail_cap;
s_ask "CREW_AVAIL ?" "yes, no" -> crew_avail, end)
ELSE (null, end)]

STATE: end
[(s_print "*********************************************************_
****
* Advice will be given based on the constraints indicated.  *
* Infeasibilities are usually caused by bad data, especially *
* when the analysis began with a feasible solution.        *
*                                                          *
* You will be provided with information on the constraint  *
* and the input data that might be causing the infeasibilty. *
**********************************************************";analysis,
GOAL)]
```

```
/*  FILE: analysis.ind  */

MODULE: infeasible.prelim.constraints.analysis
DECLARATIONS:
[
        INTENT: "giveslappropriate advice for indicated constraints|"
        CHILD: supp_and_demand
        CHILD: sgroup
        CHILD: crew_data
        CHILD: flows
        CHILD: conflow_bnd
        CHILD: genflow_bnd
        CHILD: imp_frac_lim
        CHILD: tline_load
        CHILD: production
        ]

STATE: hydro
[IF hydro_pro IS
"yes": (production;s_adv3
"--> Check for an error in the:
MINIMUM CAPACITY REQUIREMENT
of the power plant indicated."
"check for data entry errors for this value"
"Look in the input data file under the plant
number, i.e. number in the list.", import_frac)
ELSE (null, import_frac)]

STATE: import_frac
[IF fraction IS
"yes": (imp_frac_lim, fbound)
ELSE (null, fbound)]

STATE: fbound
[IF genflow IS
"yes": (genflow_bnd, conbound)
ELSE (null, conbound)]

STATE: conbound
[IF conflow IS
"yes": (conflow_bnd, stripgroup)
ELSE (null, stripgroup)]

STATE: stripgroup
[IF strip_group IS
"yes": (s_adv3
"A strip grouping constraint was violated.
--> You can temporarily remove this strip group from the problem and
then rerun it to see if it is the cause. Then check the indicated strip
group data for any errors. If data is correct the group can likely not
fulfill requirements placed on it."
"check for data errors in the strip group input data"
```

"Further advice will be provided";
sgroup;
s_adv "May want to do a hand check on the group:
strip intercept + slope*tstep load >= plant memberships * plant capacity", maintenance)
ELSE (null, maintenance)]

STATE: maintenance
[IF maint_con IS
"yes": (null, maint_check)
ELSE (null, energy)]

STATE: maint_check
[IF (s_ask "Were other constraints indicated along with the maintenance ones?" "yes, no") IS
"yes": (s_adv3
"You may wish to set the MAINT_ALLOWED flag to NO and
--> rerun the program to see if the infeasibility was caused by the maintenance requirements of the plants or some other factor. If this does not solve the problem then do a quick check of the maintenance scheduled for the plants in the input data."
"EMMA can run the system with or without
maintenance considerations"
"MAINT_ALLOWED is set in the NETWORK STRUCTURE DATA
section of the input data file" ,energy)
ELSE (crew_data, GOAL)]

STATE: energy
[IF supply IS
"yes": (s_print "An energy requirement constraint has been violated.";
supp_and_demand, flow_check)
ELSE (s_adv3
"Since the SUPP_AND_DEMAND constraints were not indicated,
--> the energy requirements of the system are not involved in the infeasibility. The capacity in the system is able to meet demands.
The variables involved in this constraint do not need to be checked."
"know if the energy requirements are involved"
"It is often helpful to know what is not involved in the
infeasibility, especially something important like energy
requirements";supp_and_demand;
s_adv3 " It is likely a problem with flow continuity.
It may indicate errors in:
- FLOW SLOPES AND INTERCEPTS of a powerhouse
- FLOW SLOPES AND INTERCEPTS of a control structure
- INFLOWS
- the values of STRIP LOAD SLOPES or INTERCEPTS
--> The analysis will be continued."
" "
"Flow continuity is often violated in an infeasible run.
Further information is needed to provide more useful advice.", flow_check)]

STATE: flow_check
[IF flow_cont IS
"yes": (s_print "Flow continuity has been violated.";
flows; s_adv3 "Flow continuity problems will continue downstream of source. Therefore, the lake data for the first lake indicated in the first violated FLOW_CONTIN constraint should be checked. The same can be said about the time step indicated in this first indicated constraint. This will tell when the flow problem began.
--> Check the inflows into this lake and the stage-storage slopes and intercepts and stages set for this lake. If it was the only constraint indicated then check the inflows to this lake and minimum flow slopes and intercepts set on any control structure that might exist on this lake."
"explain the reaction of a water system to flow
continuity problems"
"Data checks should be limited to the cause of the
actual flow problem, not all the resulting effects", ties)
ELSE (s_print "The flow continuity constraints have not been violated,
--> so the problem is with the energy requirements or maintenance.
Any input variables involved in this constraint need not be checked.";flows;
s_adv3 "Since flow continuity is not involved there may be an error in one of the following areas:
- CAPACITY DERATES assigned to power plants
- STRIP LOAD SLOPE
- calling a plant BASE_LOADED when it has the data structure
of a SHAPED generation pattern."
"know if flow continuity is not involved"
"It is useful to know that flow continuity is NOT
involved in the infeasibility. The advice can be
more specific", ties)]

STATE: ties
[IF tie_load IS
"yes": (s_print "A tieline constraint has been violated.";
tline_load;
s_adv3 "--> Since this constraint is indicated, the suggestion is to check for an incorrect setting for:
MAXIMUM FRACTION IMPORTS (set in the load data)
IMPORT and EXPORT capacity
of the indicated tieline"
"advise user of data checks to make if tieline load
constraints are vilated"
"Two possible data checks are listed.", GOAL)
ELSE (null, GOAL)]

```
/*  FILE: production.ind  */

MODULE: infeasible.prelim.constraints.analysis.production
DECLARATIONS:
[INTENT: "providelinformation on effect of hydro production constraintl"]

STATE: one
[(s_adv3
"A hydro production constraint has been violated.
HYDR_PROD(gstation, tstep)
and is created to convert powerhouse discharge into energy.
It involves the GENFLOW through a powerhouse."
"describe the hydro production constraint"
"This constraint is not very helpful in
pinpointing possible infeasibility causes", GOAL)]
```

```
/*  FILE: imp_frac_lim.ind  */

MODULE: infeasible.prelim.constraints.analysis.imp_frac_lim
DECLARATIONS:
[INTENT: "adviselabout possible errors with import restrictionsl"]

STATE: import_fraction
[(s_print "The following constraint has been violated:
IMP_FRAC_LIM(tstep)
and is used to limit the total interruptible (not firm)
import energy to a percentage of the required system
energy for a time step.
It involves the following input data:
SYSTEM ENERGY
FIRM IMPORTS
IMPORT FRACTION LIMIT";
s_adv3
"--> The cause of the infeasibility is likely an incorrect
value specified for the maximum fraction allowed from
imports as specified in the load curve data. If the number
is correct, the system cannot meet the enegy requirements
unless it is allowed a greater percentage from interruptible
imports."
"descibe the constraint restricting imports"
"Look at input file in the LOAD CURVE DATA section.", GOAL)]
```

```
/* FILE: genflow_bnd.ind */

MODULE: infeasible.prelim.constraints.analysis.genflow_bnd
DECLARATIONS:
[INTENT: "adviselabout possible errors in genflow specificationsl"]

STATE: one
[(s_print "A flow bound at a generating station has been violated.
The constraint is of the form:
GENFLOW_BND(1=powerhouse or 2=spillway, gstation, tstep)
and is used ot limit the outflow from a powerhouse or spillway as a
function of upstream storage.

The variables involved inthis constraint are:
LAKE STORAGE
POWERHOUSE/SPILLWAY FLOW INTERCEPTS AND SLOPES";
s_adv3
"--> The suggestion is to check the values for:
FLOW INTERCEPTS and SLOPES of the POWERHOUSE or SPILLWAY
for the hydro generating station identified."
"describe the genflow constraint"
"Infeasibility may be due to setting very restrictive
max and min flows at the power stations.", GOAL)]

/* FILE: conflow_bnd.ind */

MODULE: infeasible.prelim.constraints.analysis.conflow_bnd
DECLARATIONS:
[INTENT: "adviselabout possible control structure data errorsl"]

STATE: one
[(s_print "A control flow constraint has been violated and has the form:
CONFLOW_BND(control structure, tstep)
and is used to limit the outflow from a control structure as a function of
upstream storage.

The input data involved in this constraint are:
LAKE STORAGE
CONTROL STRUCTURE FLOW SLOPES AND INTERCEPTS";
s_adv3
"--> Check for errors in the specifications for:
SLOPES and INTERCEPTS for the CONTROL STRUCTURE(S)
indicated in the violated constraints."
"describe the control structure flow constraint"
"Flow slopes and intercepts control max and min
flows out of control structures.", GOAL)]
```

```
/* FILE: sgroup.ind */

MODULE: infeasible.prelim.constraints.analysis.sgroup
DECLARATIONS:
[INTENT: "adviselabout possible errors in strip groupingl"]

STATE: one
[(s_adv3
"The constraint is of the form:
STRIP_GROUP(strip group, tstep, strip)
and is used to place a minimum (as a function of system load) on the
total energy production of this group for each strip load curve.
The input data involved in this constraint are:
MINIMUM GENERATION SLOPE
MINIMUM GENERATION INTERCEPT
LOAD CURVE DATA
MEMBERSHIP
--> Problems often arise due to the slope values,
so make sure the value is correct"
"describe the strip group constraint"
"The data are located in the STRIP GROUP section
of the file. The minimum generation may be set
too high", GOAL)]

/* FILE: crew_data.ind */

MODULE: infeasible.prelim.constraints.analysis.crew_data
DECLARATIONS:
[INTENT: "adviselabout possible maintenance data errorsl"]

STATE: one
[IF crew IS
"yes": (s_adv3
"This maintenance constraint is of the form:
CREW_SCHEDULE (crew)
and is only created if the MAINT_ALLOWED flag is set to YES.
It is used to ensure that for a specified period, crew holidays, etc. are
accounted for.
The crew input data involved in this constraint is:
MAX_PERIOD_AVAIL
--> Check the value of this variable for the indicated crew."
"describe the crew scheduling constraint"
"Crew may not be able to meet maintenance requirements
if low availability has been specified.", two)
```

ELSE (null, two)]

STATE: two
[IF reqd_maint IS
"yes": (s_adv3
"The following maintenance constraint has been indicated:
HYDRO_REQD_MAINT(gstation) or GENSRC_REQD_MAINT(gsource)
and is only created if the MAINT_ALLOWED flag has been set to YES.
It is used to ensure that sufficient maintenance at each plant is done to meet requirements.
--> Check that the proper crew has been assigned this plant. Perhaps, the crew is unable to service all the plants it has been assigned."
"describe the plant maintenance assignment constraint"
"A possible cause of the infeasibility is listed and
should be checked",three)
ELSE (null, three)]

STATE: three
[IF crew_avail IS
"yes": (s_adv3
"The following maintenance constraint was indicated:
CREW_AVAIL(crew, tstep)
and is only created if the MAINT_ALLOWED flag is set to YES.
It is used to ensure that maintenance done by a crew within a time step does not exceed their availability in hours.
--> Check that the correct value exists for:
MAX_AVAIL."
"descibe the crew availabilty constraint"
"A crew may be constrained by a low value", GOAL)
ELSE (null, GOAL)]

/* FILE: supp_and_demand.ind */

MODULE: infeasible.prelim.constraints.analysis.supp_and_demand

DECLARATIONS:
[INTENT: "adviselabout possible errors in energy generation datal"]

STATE: explanation
[(s_print "The constraint is of the form:
SUPP_AND_DEMAND(strip, time_step)
It is created to ensure that, for every load duration curve strip, system load + firm exports are supplied by genstations, gen_sources and imports.
The following input variables are involved in this constraint:
SYSTEM ENERGY i.e. load curve data
FIRM EXPORTS

FIRM IMPORTS
SYSTEM RESERVE i.e. reserve capacitiy set in the load data", one)]

STATE: one
[IF flow_cont IS
"yes": (s_adv3
"The problem could be caused by any one of the input
variables involved in this constraint.
--> The analysis will be continued."
"describe the supply and demand constraint variables"
"Since supply and demand was indicated, there are
numerous possible causes.", GOAL)
ELSE (s_adv3 "--> It is suggested that you check for errors in
- the LOAD CURVE DATA eg. loads close to peak
- the CAPACITY DERATE values for the generating stations
- that you have not specified a hydro generating station
as BASE_LOADED when it should be SHAPED."
"describe the supply and demand constraint variables"
"Since flow continuity was not involved, the
possible causes of the infeasiblity are limited.", GOAL)]

/* FILE: tline_load.ind */

MODULE: infeasible.prelim.constraints.analysis.tline_load
DECLARATIONS:
[INTENT: "adviselabout possible errors in tieline datal"]

STATE: one
[(s_print " The tieline load constraint has the form:
TLINE_LOAD(tieline, strip, tstep)
and is created to ensure that the time a tieline is importing and exporting does not exceed the total time.
The input data involved in this constraint are:
EXPORT CAPACITY of the tieline
IMPORT CAPACITY of the tieline
FIRM IMPORTS
FIRM EXPORTS", GOAL)]

APPENDIX C

Listing of Preliminary Dam Design Prototype

RUNTIME;
ENDOFF;
ACTIONS
CLS
DISPLAY "

DAM DESIGN ADVISOR

This expert system helps the user select appropriate values for:

* earthquake design criteria
* spillway design flood.

The advice given is based on work done by the National Research Council (NRC), to detail current design practices and give tentative recommendations.

Author: Kim Barlishen (5708935), U. of Manitoba, March, 1989

Press any key to begin the analysis.~"

CLS
DISPLAY "

CLASS.KBS

This module is responsible for putting a dam into the proper hazard and size classes, which are necessary to give the appropriate earthquake and flood advice.

The classification is based on the following dam properties:

* volume of water impounded
* height of dam
* damage estimates

Press any key to begin the classification process.~"
CLS

FIND size

FIND hazard

CLS
DISPLAY "

RESULTING CLASSIFICATIONS:

SIZE = {#size}

HAZARD CLASS = {#hazard}

Press any key to begin the spillway design flood analysis!~"

SAVEFACTS classes

CHAIN flood;

RULE 1
IF volume>50000
OR height>100
THEN size=large
BECAUSE "
Volume and height are required to
determine size classification of
a dam. Volume>50000 ac-ft or
height>100 ft classifies it as large.";

RULE 2
IF volume>1000
OR height>40
THEN size=intermediate
BECAUSE "
Volume and height are required to determine
the size classification of the dam, and a
volume between 1000 and 50000 ac-ft or a
height between 40 and 100 ft classifies it
as being of significant size";

RULE 3
IF volume<=1000
OR height<=40
THEN size=small
BECAUSE "
Volume and height are used to
determine the size classification of
the dam, and a volume less than 1000
ac-ft or a height less than 40 ft
classifies a dam as being small.";

RULE 4
IF loss_of_life=more

```
OR   economic_loss=excessive
THEN hazard=high
BECAUSE "
If damage estimates are high,
the dam is considered to be a
high hazard dam.";

RULE 5
IF   loss_of_life=few
OR   economic_loss=appreciable
THEN hazard=significant
BECAUSE "
If damage estimates are relatively
high, the dam is considered to be
in the significant hazard class.";

RULE 6
IF   loss_of_life=none
OR   economic_loss=minimal
THEN hazard=low
BECAUSE "
If damage estimates are relatively
low, the dam is considered to be a
low hazard dam.";


ASK volume:"What is the volume of water impounded by the dam (ac-ft)?";

ASK height:"What is the height of the dam (ft)?";

ASK loss_of_life:"What is the estimated loss of life (i.e., extent of
development) associated with failure?

        where: none  -> no permanent structures for human habitation
               few   -> no urban development and no more than a small
                        number of inhabitable structures
               more  -> more than a few
";
CHOICES loss_of_life: none,
                      few,
                      more;

ASK economic_loss:"What is the estimated economic loss associated with
failure, based on extent of development?

      where: minimal     -> undeveloped to occassional structures or
                            agriculture
             appreciable -> notable agriculture, industry, or structures
             excessive   -> extensive community, industry or agriculture
";
CHOICES economic_loss: minimal,
                       appreciable,
```

excessive;

EXECUTE;
RUNTIME;
ENDOFF;
ACTIONS
LOADFACTS CLASSES
CLS
DISPLAY "

FLOOD.KBS

This module details current design practices and gives a final recommendation (if one is available) for spillway design, based on the size and hazard classifications determined in CLASS.KBS.

No further input is required.

Press any key to continue.~"
CLS

FIND sef

DISPLAY "FINAL RECOMMENDATION FOR THE SAFETY EVALUATION FLOOD: {#sef}

Press any key to start the earthquke design advice system!~"

CHAIN earth;

RULE 1
IF size=large
AND hazard=high
THEN sef=pmf
DISPLAY "

FLOOD DESIGN CRITERIA

Most federal agencies use PMF.
State agencies and firms also tend to use PMF.

Other stated criteria are:
* 0.4PMP, PMP
* 0.5PMF, 0.5PMF to PMF
* 1000 yr. flood.

"
BECAUSE "
Spillway flood design criteria are
related to size and hazard classes.";

RULE 2
IF size=large
AND hazard=significant
THEN sef=not_available
DISPLAY "

FLOOD DESIGN CRITERIA

Most federal agencies use PMF.
State agencies and firms also tend to use PMF.

Other stated criteria are:
* 0.4PMP, 0.5PMP, 0.75PMP
* 0.5PMF
* 100 yr. flood to 0.5PMF
* 0.5PMF to 10000 yr. flood (larger)
* 150% OF 100 yr. flood
* P100 + 0.4(PMP - P100)

"
BECAUSE "
Spillway flood design criteria are
related to size and hazard classes.";

RULE 3
IF size=large
AND hazard=low
THEN sef=not_available
DISPLAY "

FLOOD DESIGN CRITERIA

The only specified federal criteria is from the Army
Corps of Engineers - National Dam Inspection Program
where they used: 0.5PMF to PMF.

State agencies and firms have varied criteria:
* 0.5PMF to PMF, 0.25PMF
* 0.4PMP, 0.5PMP
* 100 yr. flood, 150% of 100 yr. flood
* 10000 yr. flood, 0.3PMF or 1000 yr. flood (larger)
* P100 + 0.12(PMP - P100)

"
BECAUSE "
Spillway flood design criteria are

related to size and hazard classes.";

RULE 4
IF size=intermediate
AND hazard=high
THEN sef=pmf
DISPLAY "

FLOOD DESIGN CRITERIA

Federal agencies use PMF.
The majority of state agencies and firms use PMF.

Other stated criteria:
* 0.5PMF to PMF
* 0.3PMP, 0.5PMP, 0.75PMP, PMP
* 10000 yr. flood

"
BECAUSE "
Spillway flood design criteria are
related to size and hazard classes.";

RULE 5
IF size=intermediate
AND hazard=significant
THEN sef=not_available
DISPLAY "

FLOOD DESIGN CRITERIA

Most federal agencies use PMF, but 0.5PMF to PMF and
P100 + 0.4(PMP - P100) are also used.
The most widely used criteria in state agencies and
firms is: PMF.

Other stated criteria are:
* 0.5PMF to PMF
* 0.3PMP, 0.5PMP
* 100 yr. flood to 0.5PMF
* 0.5PMF or 10000 yr. flood (larger)
* 150% of 100 yr. flood, 10000 yr. flood

"
BECAUSE "
Spillway flood design criteria are
related to size and hazard classes.";

RULE 6

IF size=intermediate
AND hazard=low
THEN sef=not_available
DISPLAY "

FLOOD DESIGN CRITERIA

The only specified federal agency criteria is that used by the Corps of Engineers in their National Dam Inspection Program: 100 yr. flood to 0.5PMF.

Other state agencies and firms use:
* 0.25PMF
* 0.25PMP, 0.33PMP
* 100 yr. flood to 0.5PMF
* 0.3PMF or 1000 yr. flood (larger)
* 100 yr. flood, 10000 yr. flood
* P100 + 0.12(PMP - P100)

"
BECAUSE "
Spillway flood design criteria are
related to size and hazard classes.";

RULE 7
IF size=small
AND hazard=high
THEN sef=pmf
DISPLAY "

FLOOD DESIGN CRITERIA

Most federal agencies use PMF, but the Corps of Engineers in their National Dam Inspection Program used: 0.5PMF to PMF.
The majority of state agencies and firms are using PMF or 0.5PMF to PMF.

Other stated criteria are:
* 0.25PMP, 0.33PMP, 0.5PMP, PMP
* 10000 yr. flood

"
BECAUSE "
Spillway flood design criteria are
related to size and hazard classes.";

RULE 8
IF size=small
AND hazard=significant
THEN sef=not_available
DISPLAY "

FLOOD DESIGN CRITERIA

Federal agencies tend to use: PMF or
100 yr. flood to 0.5PMF
but P100 + 0.4(PMP - P100) is also used.

Other state agencies and firms are using:
* 0.5PMF, 0.5PMF to PMF
* 0.33PMP, 0.5PMP
* 0.5PMF or 10000 yr. flood (larger)
* 100 yr. flood, 225% of 100 yr. flood, 10000 yr. flood
* P100 + 0.4(PMP - P100)

"
BECAUSE "
Spillway flood design criteria are
related to size and hazard classes.";

RULE 9
IF size=small
AND hazard=low
THEN sef=not_available
DISPLAY "

FLOOD DESIGN CRITERIA

The only stated federal agency criteria comes from the
Corps of Engineers National Dam Inspection Program:
50 yr. flood to 100 yr. flood.
The most widely used criteria in state agencies and
firms is the 100 yr. flood.

Other stated criteria are:
* 0.25PMF, PMF
* 0.3PMF or 1000 yr. flood (larger)
* 100 yr. flood, 1000 yr. flood, 10000 yr. flood
* P100 + 0.12(PMP - P100)

"
BECAUSE "
Spillway flood design criteria are
related to size and hazard classes.";

EXECUTE;
RUNTIME;
ENDOFF;
ACTIONS
LOADFACTS classes
CLS
DISPLAY "

EARTH.KBS

This module is responsible for generating the appropriate advice regarding the selection of a Safety Evaluation Earthquke, design practices, and analysis procedures, that should be employed.

The analysis is based partially on the hazard classification of the dam, but it also requires the following values:

* the seismic zone classification of the area
* estimated peak ground acceleration (for earth dams)
* type of dam.

Press any key to continue the analysis.~"
CLS
FIND min_coeff
CLS
DISPLAY "

STATIC ANALYSIS ADVICE

The following is the Corps of Engineers criteria, which are typical of other agencies (giving coefficients to be multiplied by the weight of the structure to determine estimated horizontal loadings.) This value is used when performing a pseudostatic analysis.

MINIMUM COEFFICIENT = {#min_coeff}

Press any key to continue the analysis.~"
CLS
FIND design
CLS
FIND see

CLS
DISPLAY "

FINAL ADVICE:

NOTE: A safety evaluation of the dam need not consider the simultaneous occurrence of the safety evaluation flood and the safety evaluation earthquake, because of the extremely low probability associated with this.

* The flood and earthquake design advice has been given.

Press any key to end the consultation.~";

RULE 1
IF zone=0
THEN min_coeff=0g
BECAUSE "
The minimum coefficient for
earthquake forces to be used in
a stability analysis depends on
the seismicity of the area.";

RULE 2
IF zone=1
THEN min_coeff=0.05g
BECAUSE "
The minimum coefficient for
earthquake forces to be used in
a stability analysis depends on
the seismicity of the area.";

RULE 3
IF zone=2
THEN min_coeff=0.10g
BECAUSE "
The minimum coefficient for
earthquake forces to be used in

a stability analysis depends on
the seismicity of the area.";

RULE 4
IF zone=3
THEN min_coeff=0.15g
BECAUSE "
The minimum coefficient for
earthquake forces to be used in
a stability analysis depends on
the seismicity of the area.";

RULE 5
IF zone=4
THEN min_coeff=0.20g
BECAUSE "
The minimum coefficient for
earthquake forces to be used in
a stability analysis depends on
the seismicity of the area.";

RULE 6
IF zone=0
OR zone=1
AND type=concrete
THEN design=done
CLS
DISPLAY "

EARTHQUAKE DESIGN ADVICE

Among federal agencies, methods of analysis for concrete dams (in areas of low seismicity) are generally as follows:

- defensive design measures (studies to ensure foundation and abutment integrity, good geometrical configuration, effective quality control, etc.)
- pseudostatic analysis (commonly used to check sliding and overturning stability)

Press any key to continue the analysis.~"
BECAUSE "
Earthquke design advice depends
on the seismicity of the area and
the dam construction material.";

RULE 7
IF zone=2
OR zone=3

```
OR   zone=4
AND type=concrete
THEN design=done
        CLS
        DISPLAY "
```

EARTHQUAKE DESIGN ADVICE

Among federal agencies, methods of analysis for concrete dams (in areas of significant seismicity) are generally as follows:

- defensive design measures (studies to ensure foundation and abutment integrity, good geometrical configuration, effective quality control, etc.)
- dynamic analysis (assuming the dam to consist of linear elastic, homogeneous, isotropic material) to determine structural response and induced stress, since this is an area of significant seismicity.

Press any key to continue the analysis.~"

```
BECAUSE "
Earthquke design advice depends
on the seismicity of the area and
the dam construction material.";

RULE 8
IF   type=earth
AND  peak<0.2
THEN design=done
        CLS
        DISPLAY "
```

EARTHQUAKE DESIGN ADVICE

Among federal agencies, methods of analysis for earth dams (in areas where expected ground accelerations are fairly low) are generally as follows:

- defensive design measures (ample freeboard, wide transition zones, etc.)
- dynamic analysis for liquefaction potential or strain potential, for dams involving embankment or foundation soils that may lose a significant portion of their strength under the effects of earthquake shaking.
- for reasonably well built dams on stable foundation

soils, no further analysis is required.

Press any key to continue the analysis.~"
BECAUSE "
Earthquke design advice depends
on the seismicity of the area and
the dam construction material.";

RULE 9
IF type=earth
AND peak>=0.2
THEN design=done
CLS
DISPLAY "

EARTHQUAKE DESIGN ADVICE

Among federal agencies, the methods of analysis for earth dams (where peak ground acceleration is fairly high) are generally as follows:

- defensive design measures (ample freeboard, wide transition zones, etc.)
- dynamic analysis for liquefaction potential or strain potential, for dams involving embankment or foundation soils that may lose a portion of their strength as a result of earthquake shaking.

Press any key to continue the analysis.~"
BECAUSE "
Earthquke design advice depends on
the seismicity of the area and
the dam construction material.";

RULE 10
IF hazard=high
THEN see=found
DISPLAY "

SAFETY EVALUATION EARTHQUAKE

For high hazard dams, the safety evaluation earthquke should be:

- the maximum credible earthquake (MCE) with ground

motions developed by the deterministic-statistical method applied to known causitive faults. This should provide and acceptable level of conservatism for safety analysis.
- where earthquake sources are not well identified, the seismotectonic province method for determining MCE should be used.

Press any key to continue the analysis.~"
BECAUSE "
Advice regarding a suitable safety
evaluation earthquake depends
on the hazard class of a dam.";

RULE 11
IF hazard=significant
OR hazard=low
THEN see=found
DISPLAY "

SAFETY EVALUATION EARTHQUAKE

For dams in the significant and low hazard classes, there is no specific recommendation for the value of the dam safety evaluation earthquake.

However, a less severe value than the maximum credible earthquake (MCE) can be used.

The MCE can be developed using the deterministic-statistical method applied to known causitive faults. If the earthquake sources are not well identified, the seismotectonic province method for determining MCE should be used.

Press any key to continue the analysis.~"
BECAUSE "
Advice regarding a suitable safety
evaluation earthquake depends
on the hazard class of a dam.";

ASK peak:"What is the estimated peak ground acceleration (Xg) for this area?";

ASK type:"What type of dam is it?";
CHOICES type: concrete,
earth;

ASK zone:"What is the seismic zone classification for the area?";
CHOICES zone: 0, 1, 2, 3, 4;