1992

Application of expert system techniques for award implementation

Seng-Choon Yeoh
Edith Cowan University

Edith Cowan University

Copyright Warning

APPLICATION OF EXPERT SYSTEM TECHNIQUES FOR AWARD IMPLEMENTATION

By

Seng-Choon Yeoh

A Thesis Submitted in Partial Fulfilment of the Requirements for the Award of

Bachelor of Applied Science (Information Science) Honours

at the

Faculty of Science and Technology

School of Information Technology and Mathematics

Department of Computer Science

Edith Cowan University

Perth, Western Australia

4th December, 1992

USE OF THESIS

The Use of Thesis statement is not included in this version of the thesis.

DEDICATION

In treasured memory of two special people in my life:

Jens Rainer Geldermann

18.Nov.1969 - 11.Oct.1992

My best and greatest mate.

Laurence Oon Huat Yeoh

23.Dec.1943 - 31.Aug.1990

My best and greatest uncle.

And to the Yeoh Dynasty

My Kingdom of Jewels.

--ooOoo--

"Man cannot discover new oceans until he has courage to lose sight of the shore."

[Anon.]

ABSTRACT

This paper describes a pilot expert system project, PESWEA, developed in conjunction with The Western Australian Government Railways Commission (Westrail), to locate and implement an expert system pilot solution to the problems and complexities of the awards. The paper will outline the process used to develop and implement a solution to a business requirement. In particular, the methodologies for the project, shell selection, and knowledge acquisition will be presented and discussed.

The fundamentals of expert systems will be discussed to provide the reader an insight into the technology. In addition, this paper reviews the literature relevant to the research questions. Empirical findings in the literature are discussed and analysed to discover how they influence the work in this paper. The topics covered include the concepts of expert system shell selection, knowledge acquisition and representation, and the integration of expert systems and database systems. A case study on a similar pilot system conducted by the State Electricity Commission of Victoria (SECV) is also reviewed.

The PESWEA knowledge-bases were implemented with the 1st-Class HT expert system shell which succeeded in meeting the selection criteria for the first study in this project. The work carried out has also confirmed that the expert system techniques can be used to gather and interpret information from manual time-sheets which are subject to a complex arbitration award. Furthermore, PESWEA utilised the Inter-system Communication approach to systems interfacing, with the DataFlex database system dominating the concentration of processing and control.

The technical and business objectives of PESWEA have been achieved with success. This study has confirmed SECV's own research into the applicability of expert system techniques for award implementation. From this study, Westrail and Edith Cowan University believe that expert systems technology can now be integrated into the mainstream programming techniques at Westrail.

DECLARATION

" I certify that this thesis does not incorporate, without acknowledgment, any material previously submitted for a degree or diploma in any institution of higher education and that, to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where due reference is made in the text".

Signature:_

Date: 4th Dec. 1992.

ACKNOWLEDGMENT

I wish to express my gratitude to my supervisor Dr Chaiyaporn Chirathamjaree and CEED coordinator Robert Cross for their invaluable support and guidance throughout the preparation of my project and thesis; to Lindsay Burke and the staff at the I.T. branch of Westrail for their faith, enthusiasm, and support; special thanks to Vincent Phillips for his editing skills - please direct all grammar criticisms to him; to fellow Computer Science postgraduate students for their advice, encouragement and entertainment - without them our Postgraduate Lab wouldn't be a zoo; and the support of all at the Faculty of Science and Technology, Mt Lawley and Bunbury campuses.

Most of all I would like to thank my parents for their love, encouragement, advice, patience and never-ending support throughout my long years of education, and finally to my brother (*yo bro!*) for showing me the other side of life.

For those curious about such things, this thesis was written with a WordPerfect for Windows 5.1 word-processor on a 386DX compatible computer. The figures were designed using DrawPerfect 1.1 and imported into the document with WordPerfect's graphics facilities. The thesis was printed on a Hewlett-Packard LaserJet IIP Postscript printer.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Chapter Headings:

1.1 INTRODUCTION

This paper describes a knowledge base systems pilot, PESWEA (Pilot Expert System for Westrail Employee Awards), developed in conjunction with the Western Australian Government Railways Commission (Westrail). This paper investigates the technical feasibility of applying expert system technology to the task of gathering information from manual timesheets which are subject to a complex arbitration award.

1.1.1 Background

This project was aimed at applying expert system technologies to the complexities of the Westrail employee awards. The application of such technology to awards is still a relatively unknown concept. Only one prototype has recently been published in Australia to assist with the interpretation of the numerous award conditions. At the time of publication (Plant, Smalley & Waterson, 1990) the prototype system was thought to be unique in the world and its success as a pilot attracted the interest of numerous departments of the Australian government.

The timesheet processing department at the East Perth Westrail Centre was selected as an appropriate application area for the organisation's first serious effort in applying expert systems technology. The department offered an appropriately sized problem with a number of enthusiastic experts and a library of well-documented award manuals.

1.2 APPLICATION DESCRIPTION

1.2.1 Timesheet processing description

As a large organisation, Westrail employs thousands of workers under many different awards based on their job categories. The vast majority of Westrail's work force is employed under the conditions of three major awards, each of which is further segregated into numerous sub-awards or groups which embraces employees of the various branches. The awards serve to dictate the working and pay conditions of all employees of Westrail.

The current Westrail timesheet processing system has an annual capacity of tens of thousands of recorded timesheets. Figure 1.1 provides a schematic view of a current sample timesheet process. An off-shift worker forwards a time-card of hours lodged on duty to a field timekeeper. The field timekeeper proceeds to enter details of the time-card onto an official timesheet including appropriate penalties and allowances accumulated by the worker. The completed timesheet is forwarded to a supervisor whose task is to verify and endorse the timesheet. Batches of endorsed timesheets are sent daily to the timesheet processing department at the East Perth Westrail Centre where department timekeepers manually record timesheets into the payroll computer system.

Though effective, the current system is predominately batch-based and, in many instances, inefficient. Westrail[1] plans to streamline the entire timesheet process to a suite of online systems.

---

[1] The term *Westrail* will be consistently used to define management practises, the management themselves or the organisation as a whole.

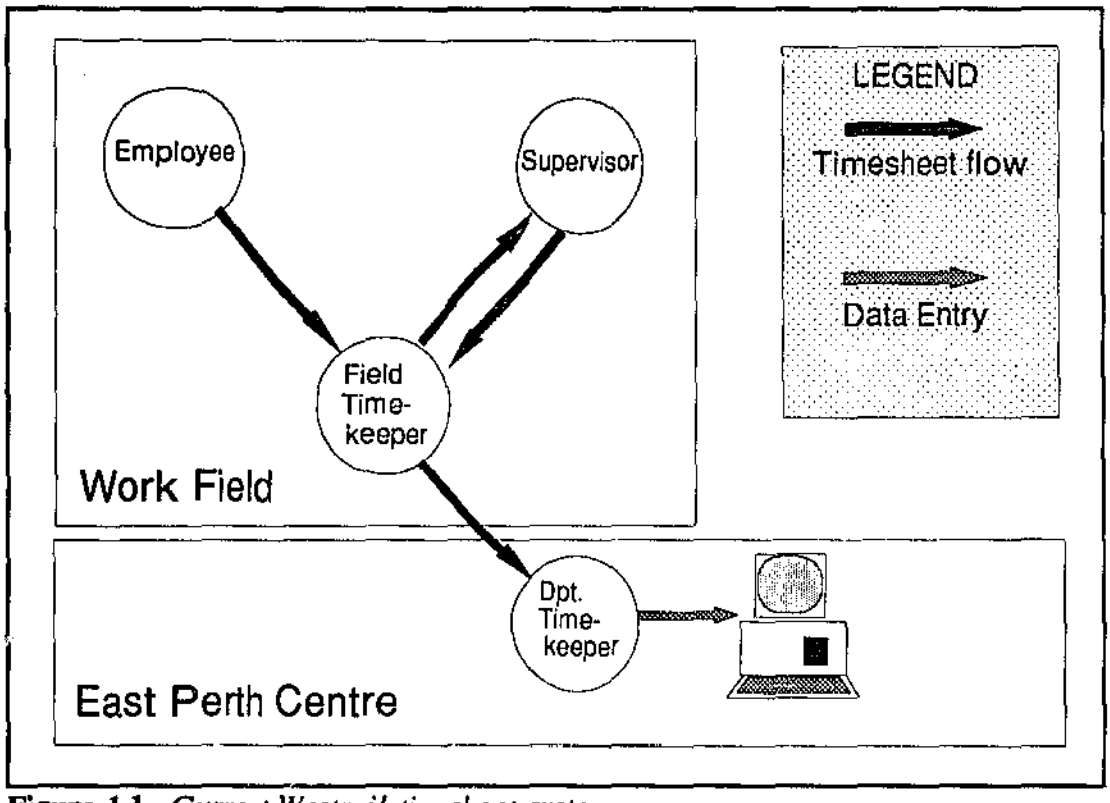

Figure 1.1 *Current Westrail timesheet system*

1.2.2 Role of PESWEA

Part of the philosophy in the introduction of the online systems is the devolvement of control to the supervisor level, and the capture and access at source of more timely and accurate management information.

The PESWEA project hopes to provide the first step towards building on the above philosophy by examining the potentials of applying expert system technology to the complexities of the award structure.

1.2.3 Positive features of the timesheet application

The development of expert systems for commercial profit, rather than purely technical interest, means that the selection of a suitable application is critically important (Coleman, 1989; Bowerman & Glover, 1988). Some features of the timesheet application that have a positive impact on the success of this project are summarised as follows:

1) *Supportive management.* Supportive management at the timekeeping and IT[2] departments gave their best to ensure that the project would not suffer from obstacles.

2) *Future projects.* This project is a stepping stone to future Westrail projects.

3) *Education value.* The pilot project was considered as a valuable learning tool for all involved.

4) *Sufficiently complex to be a serious demonstration of the technology.* The timekeeping application was considered to be the right size to apply expert system technology.

5) *Abundant case data.* In order to evaluate the success of the system there has to be sufficient case data. The award interpretations provide an abundant supply of case data to implement into the system to determine the successful outcome of applying expert system technology.

---

[2] Information Technology.

1.3 THE POTENTIAL OF EXPERT SYSTEMS IN THE AWARD DOMAIN

Expert system technology is a computer-based system that uses knowledge, facts, and reasoning techniques to solve problems that normally require the abilities of human experts (Martin & Oxman, 1988, p.14). The PESWEA project attempts to examine how expert systems can adapt to the complexities of the awards, conditions and the classification structure.

1.3.1 Class of problems

The award conditions are a highly complex set of instructions that have been carefully documented in volumes of award manuals. Only a small handful of specialised personnel or experts have the experience and knowledge of the award conditions and classification structure. These aspects are common to the award problem domain:

1) Trainees have to undergo a timekeeping correspondence course before becoming a qualified timekeeper.

2) Software engineers have a daunting task of familiarising with the awards during system development.

3) Changing awards require extensive modifications of existing programs. Likewise, new awards are expensive and time consuming to *hard-code* with conventional programs.

Harmon and King (1985, p.25) states that if the "task performance depends on knowledge that is subjective, changing, symbolic or partly judgemental, the domain may very well be a good candidate for an expert system". The award problem domain therefore, appears to be suitable for expert system development.

1.3.2 Cost advantages

When contemplating whether to explore and exploit new technologies, the primary aim of organisations is determining the effect on their *bottom line* (Crofts, Ciesielski, Molesworth, Smith & Lee 1989; Fehsenfeld, 1988). For expert systems to be feasible in the problem domain, their use must be capable of reducing existing costs (Benchimol, Levine & Pomerol, 1987). Potential cost advantages include:

1) Much faster prototyping and development time when using expert system technology for appropriate tasks. (Crofts et al., 1989; Coleman, 1989).

2) Devolvement of control to key personnel thus reducing the number of staff required for a task.

3) Building a platform to explore and exploit other system building technologies which, in the long term, may produce cost benefits.

1.3.3 System maintenance advantages

There are maintenance advantages with expert systems as opposed to traditional applications programming. If the knowledge for a specified domain was static for an expert system, it could optionally be *hard-coded.* However some problem domains are not static (for example, the diagnoses of diseases, their prognosis and their treatment), therefore the knowledge base must be able to be edited and be readily expandable. Expert systems provide such an ability as the knowledge (e.g. rules or frames) and the control mechanism (inference engine) are separate. In contrast to conventional applications software, any changes in the knowledge requires modifications in the control code (Crofts et al., 1989; Hayes-Roth, Waterman & Lenat 1983).

1.4 THE APPROACH OF THE STUDY

1.4.1 Objectives

Technical objectives. The technical objectives of PESWEA are summarised as follows:

1) demonstrate that the award conditions could be captured and applied using currently available expert systems technology.

2) demonstrate that the knowledge base has the capacity to interface with external systems.

Business objectives. The business objective was to prove the viability of expert system technology. The aim was to demonstrate to Westrail that expert systems technology could prove to be a strategically important investment opportunity.

1.4.2 Methodology

Several papers on the development of expert systems exist and will be discussed in Chapter 3. The *waterfall* life-cycle approach was selected as a model for expert systems development because, (a) the model is extensively discussed in the literature, and (b) the model has been used and proven in the expert systems industry (Guida & Tasso, 1989; Martin et al., 1988).

1.5 SCOPE OF THE STUDY

1.5.1 Research questions

The intention of the pilot project was two-fold. Though the scope of this study was restrained to the pilot project only, Westrail could use the project to further their own research into future systems. This study sought answers to three primary questions:

1. What expert system shell is most suitable to the pilot application?
2. What are the techniques to construct an expert system knowledge base and rules for award implementation?
3. How to interface the shell as a logic engine to a database system?

1.5.2 Limitations of the study

The scope of the study was delimited by the following factors:

1. The size of the pilot project was reduced to ensure satisfactory completion within the limited time frame.
2. While acknowledging the assistance offered and provided by Westrail, this study was essentially a one-person project implementation.
3. The project was sized to satisfy the objectives.

1.6 STRUCTURE OF THESIS

This thesis is segregated into five chapters:

- Chapter 2 forms the theoretical framework behind expert systems. Fundamental issues are covered to provide the reader with an insight into expert systems technology.

- Literature of empirical investigations are reviewed in Chapter 3.

- Chapter 4 covers the relevant research methodologies for the study.

- Results of the investigation are covered in Chapter 5 including the requirements of the shell based on an analysis of the requirements for the pilot system.

- And finally Chapter 6 summarises the study and conclusions, including an insight to future directions for the project.

CHAPTER 2

THEORETICAL FRAMEWORK

Chapter Headings:

2.1 EXPERT SYSTEM FUNDAMENTALS

2.1.1 What is an Expert System?

The expert systems paradigm investigates methods and techniques for constructing man-machine systems with specialised problem-solving expertise. Expertise consists of knowledge about a particular domain, understanding of domain problems, and skill at solving some of these problems (Hayes-Roth et al., 1983, p.5). Knowledge in any specialty is classed into two forms: public and private, whereby public knowledge includes published texts and references. However, human experts possess knowledge that is not made public through published text and the like, called private knowledge. Private knowledge consists of rules of thumbs known as *heuristics* (Hayes-Roth et al, 1983; Martin et al., 1988; Parsaye & Chignell, 1988). With heuristics, a human expert can make educated guesses when needed, identify approaches to problems, and dealing with incomplete or erroneous data. The fundamental task of expert systems is elucidating and reproducing such knowledge (Hayes-Roth et al., 1983; Gevarter 1990a).

Welbank (in Hart 1989, p.21) defines an expert system (which is a field of artificial intelligence) as follows:

> An expert system is a program which has a wide base of knowledge in a restricted domain, and uses complex inferential reasoning to perform tasks which a human expert could do.

2.1.2 Comparisons: Expert Systems and Conventional Programs

Expert systems and conventional programs require different development approaches. Potential developers of expert systems must first familiarise themselves with the concept of knowledge engineering (Williams, 1990, p.2). In the course of conventional computer programming, the theory of *semantics* and *syntax* of a language is emphasized (Bielawski & Lewand, 1988, p.20).

The concept behind conventional programs is to define appropriate data structures and procedures into code to solve a particular problem. During the design process, the situations to be considered, the decision points and appropriate responses are all identified beforehand. To solve a problem then, conventional programs typically operate on a *complete set of data* with the expectation of generating the *unique solution* (Bielawski et al., 1988; Williams, 1990).

The top-down development approach used for conventional programs minimises changes needed later during the stages of coding. Because of this, the top-down approach is crucial to conventional software development as changing system design becomes difficult, causing delays and escalating development costs, once coding has commenced (Williams, 1990, p.3). Where the top-down approach fails is during maintenance, where unforeseen changes and enhancements surface during application usage in the field, resulting in high software maintenance costs (Williams, 1990, p.3).

The development of expert systems is never a linear process (Bielawski et al. 1988; Williams 1990). Expert systems must encode heuristic forms of knowledge which includes symbols, strategies and relationships instead of the hard facts and rules of conventional programs. Because of this, development of expert systems is often blurred with the knowledge not becoming evident until the human expert's knowledge is entered into the system. Expert systems achieve increasing expertise in following iterations based on repeated interviews with human experts by the knowledge engineer (Bielawski et al. 1988; Kinnucan 1988; Williams 1990). Table 2.1 (Bielawski et al. 1988, p.22) summarises the basic differences between conventional programs and expert systems.

CONVENTIONAL PROGRAM	*EXPERT SYSTEM*
Requires a complete set of data.	Can function with an incomplete set of data.
Uses algorithms.	Uses heuristics or rules of thumb.
Produces a unique solution.	May produce several solutions.
Generates results that are certain.	May generate uncertain results.
Lends itself to a top-down approach to development.	Accommodates a bottom-up development methodology

Table 2.1 *Differences between conventional programs and expert systems*

2.1.3 Limitations of Expert Systems

While expert systems technology provides many benefits, the technology is still in its infancy and debate still continues over the effectiveness and its role in the marketplace and industry (Bowerman et al., 1988, p.16). The following points highlights some of the limitations of current expert system technologies:

1. *Cognitive vs. other human tasks.*

Expert systems can only perform in the domain of abstracted, logical thinking processes and are generally not able to perform complex sensory input or mechanical output without specialised interfaces. Unless good interfaces can be defined, expert systems are unable to mimic human experts to perform manual tasks beyond its cognitive reasoning (Bowerman et al., 1988; Collins, 1990).

2. *Limited vs. general intelligence simulation.*

Expert systems display a limited scope of simulated intelligence based on a specified heuristic knowledge or a single task. The intelligence of the system depends on the knowledge of the human expert, and it cannot tackle broad, multiple-direction problem spaces. As yet, it is not possible to transfer the actual intelligence of a human expert to expert systems (Bowerman et al., 1988; Collins, 1990; Buchanan & Smith, 1989).

3. *Error anomaly and recovery.*

Expert systems are only as reliable as their embedded knowledge and generally do not respond effectively to circumstances beyond their expertise (Bowerman et al., 1988; Gillies, 1991).

4. *Common sense knowledge and qualitative simulation.*

Judgment and common sense knowledge of human experts based on environmental influences including social surroundings, feelings, emotions and other non-rational information is difficult to codify into expert systems even if the influences are understood. Humans can also reason with inexact, context-sensitive concepts resembling large, small, near, far and almost. Many expert systems of today are unable to simulate these factors (Bowerman et al., 1988; Keim & Jacobs, 1986).

5. *Intuition.*

Humans have a deeper level of judgment value known as intuition which has subtle affects on the decision making and the knowledge available to the individual. It is this intelligence that is typically beyond the reach of present expert system technology (Bowerman et al., 1988; Collins, 1990).

6. *Learning.*

Although some adaptive systems are capable of acquiring knowledge and modifying behaviour which duplicates learning, it is not comparable to the ease with which humans learn from experience. Many learning systems learn from the input of the user or the knowledge engineer (Bowerman, 1988; Buchanan et al., 1989).

7. *Self-knowledge.*

Expert systems have little or no self-knowledge, and as a result do not have a sense of what they do not know. Although explanations are given of what it knows, expert systems do not have a general sense of awareness of their own knowledge (Bowerman, 1988; Buchanan et al., 1989).

2.1.4 Advantages of Expert Systems

The advantages of expert systems is best summarised by Townsend (1987, p.116):

- Expert systems can be used to solve problems when no procedure exists and the problem is unstructured.

- Expert systems are often cost effective when human expertise is very expensive, not available, or contradictory.

- Expert systems can apply a systematic reasoning process with a very large knowledge base that is often much larger than a human expert can retain or utilise.

- The expert system is objective. It is not biased or prejudiced to a predetermined goal state, and it does not jump to conclusions.

- Expert systems are not influenced by perceptions that are not relevant. The human expert's decision is easily influenced by knowledge and perceptions not directly related to the problem. The expert system's decision is related strictly to the knowledge in the knowledge base.

2.1.5 Types of Expert Systems

The terms *knowledge-based systems* and *expert systems* are interchangeable (Martin et al., 1988, p.2), and the types of applications include the following common forms. Table 2.2 (Hayes-Roth et al., 1983, p.14) gives a summation of the types of expert system applications in use today.

Diagnosis. This is the most common application of expert systems for personal computers. Its function is to analyse the symptoms of observables and identifying the associated causes. Usage includes medical illness, mechanical failures and electronics among others (Hayes-Roth et al., 1983; Townsend, 1987).

Interpretation. Interpretive systems analyse observables to determine its meaning. Common usage includes surveillance, chemical analysis, image analysis, speech understanding and other forms of intelligence analysis (Hayes-Roth et al., 1983; Townsend, 1987).

Prediction. Prediction systems serve to deduce likely consequences from a given situation and are used in meteorology to predict weather. Other usage includes military forecasting, traffic predictions and demographic predictions (Hayes-Roth et al., 1983; Townsend, 1987).

Monitoring and Control. Monitoring systems play a crucial role in many situations where observations are compared to factors that determine its successful outcome. Situations may include nuclear power plants, air traffic control and patient monitoring where quick decisions are needed based on the available data. These systems function in real-time (Hayes-Roth et al., 1983; Townsend, 1987).

Planning. Planning systems design actions that assists in the planning process. These systems can support counselling, project planning, military tactics and automatic programming among others (Hayes-Roth et al., 1983; Townsend, 1987).

Instruction. These systems can be used as a teaching aid by constructing hypothetical situations based on the student's knowledge. Instruction systems then diagnose weaknesses in the student's knowledge and concludes with appropriate remedies (Hayes-Roth et al., 1983; Townsend, 1987).

CATEGORY	PROBLEM ADDRESSED
Interpretation	Inferring situation descriptions from sensor data.
Prediction	Inferring likely consequences of given situations.
Diagnosis	Inferring system malfunctions from observables.
Design	Configuring objects under constraints
Planning	Designing actions.
Monitoring	Comparing observations to plan vulnerabilities.
Debugging	Prescribing remedies for malfunctions
Repair	Executing a plan to administer a prescribed remedy.
Instruction	Diagnosing, debugging, and repairing student behaviour.
Control	Interpreting, predicting, repairing, and monitoring system behaviours.

Table 2.2 *Categories of expert system applications.*

2.2 EXPERT SYSTEM ARCHITECTURE

2.2.1 The Structure of an Expert System

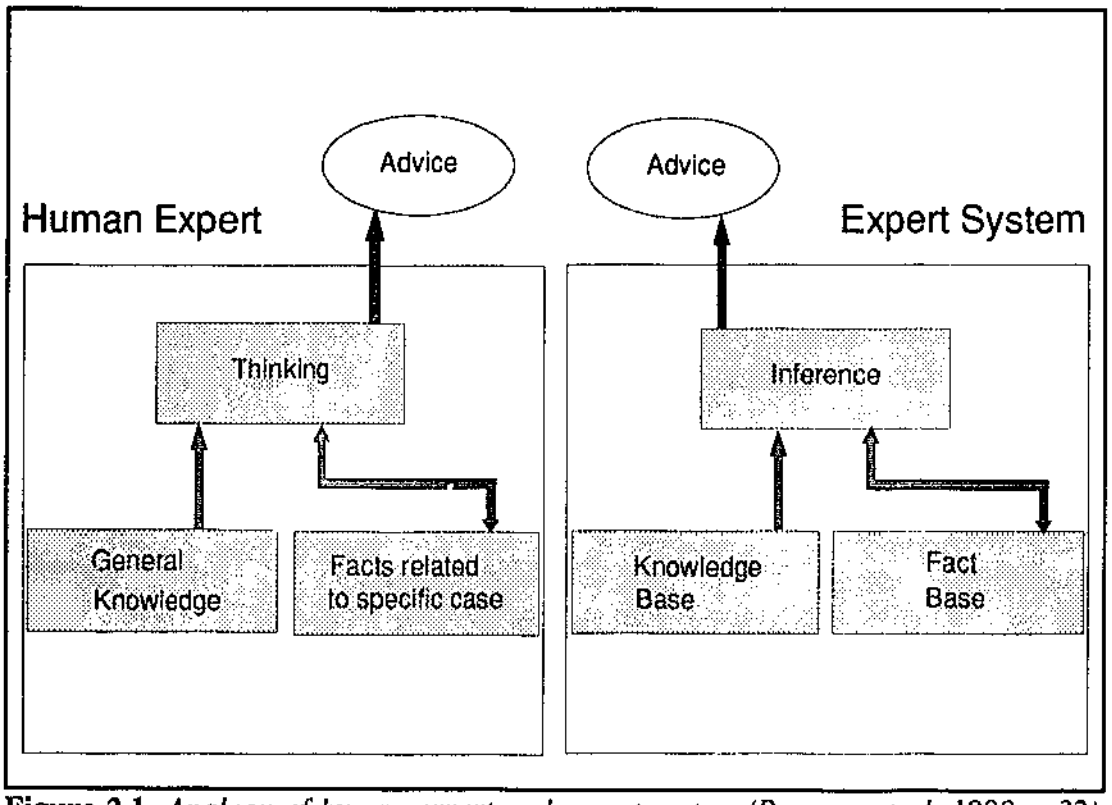

Figure 2.1 *Analogy of human expert and expert system (Parsaye et al. 1988, p.32)*

Expert systems can be compared to human experts in many ways. A human expert uses knowledge and reasoning to derive a conclusion. Likewise, expert systems rely on their knowledge and perform reasoning to arrive at conclusions (Parsaye et al., 1988, p.32). This analogy between human experts and expert systems is shown in Figure 2.1.

An expert system consists of a knowledge base, an inference engine and a working memory (Gevarter 1990a; Gevarter 1990b). The knowledge base consists of domain facts and the heuristics associated with the problem. The inference engine, or control structure, utilizes the knowledge base to find a solution to a problem. Thus, the inference engine is the part of the expert system that performs the reasoning process (Parsaye et al., 1988; Gevarter 1990a). The working memory of an expert system, or global database, is used to keep track of the problem status, input data and the working history of what has been achieved so far (Gevarter 1990a, p.18).

A variety of tools are used to interact with an expert system, which includes connections to external databases or real time data, or it may be embedded as a module in large applications (Parsaye et al. 1988, p.33). Interfaces to expert systems include the end-user interface, or external systems interfacing to an array of electronic sensors for communicating with the *relevant world*[3] (Gevarter 1990b, p.34).

The end-user interface is an important part of an expert system if its interaction relies on the user. The user interface permits users to query the system, receive advice and supply information needed by the expert system (Parsaye et al. 1988, p.33). All forms of communications between the user and the expert system are handled by the interface, hence the presentation of information to the user should comply to the user's expectations and familiarity with the task. This is known as *cognitive compatibility* (Parsaye et al. 1988, p.33).

[3] A *relevant world* is a term used to describe external systems in contact with expert systems. Expert systems gather data from these systems as opposed to users.

Often expert systems need to justify and explain their reasoning and actions. The *explanation facility* of an expert system assures the users of its actions when in use and serves as an aid to the developer during testing and development of the system (Parsaye et al. 1988, p.34).

The structure of an expert system is presented in Figure 2.2 which includes methods for building and updating the knowledge base, the inference engine and the user interface.

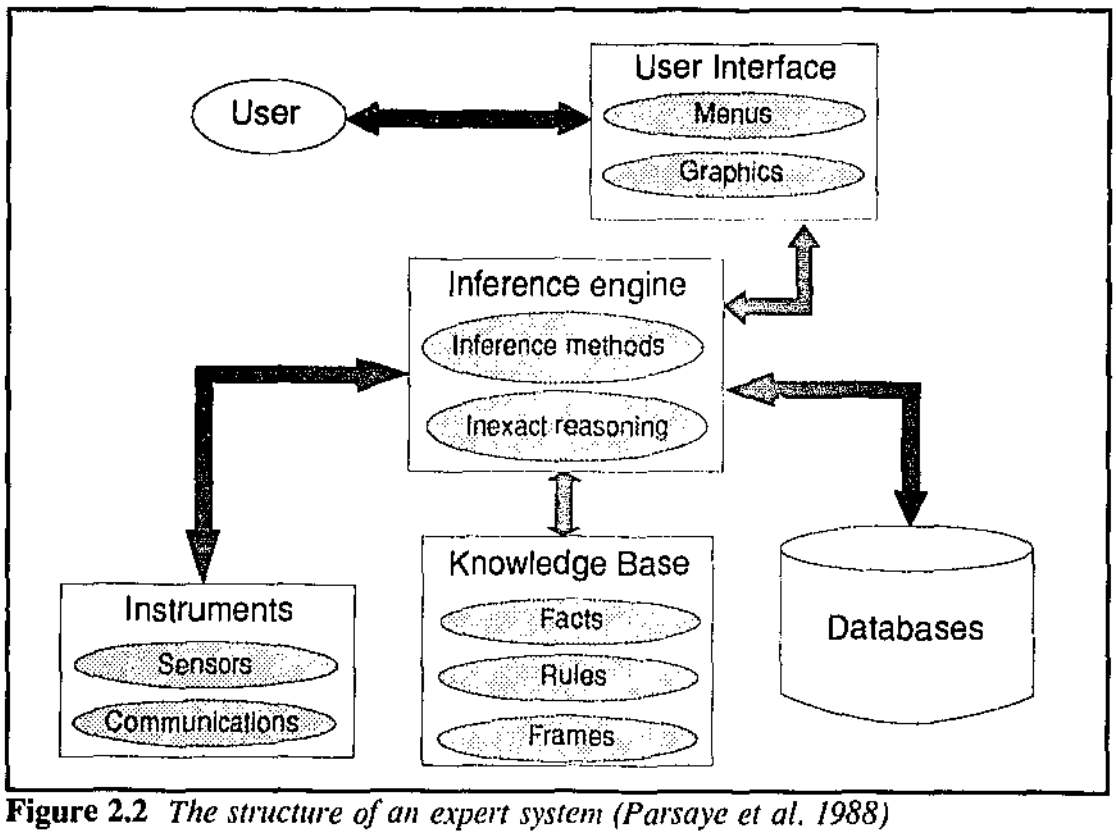

Figure 2.2 *The structure of an expert system (Parsaye et al. 1988)*

2.2.2 The Knowledge Representation

There are three fundamental forms of knowledge representation of a knowledge base: object descriptions, certainties, and actions (Gevarter 1990b, p.35). These are represented in Figure 2.3.

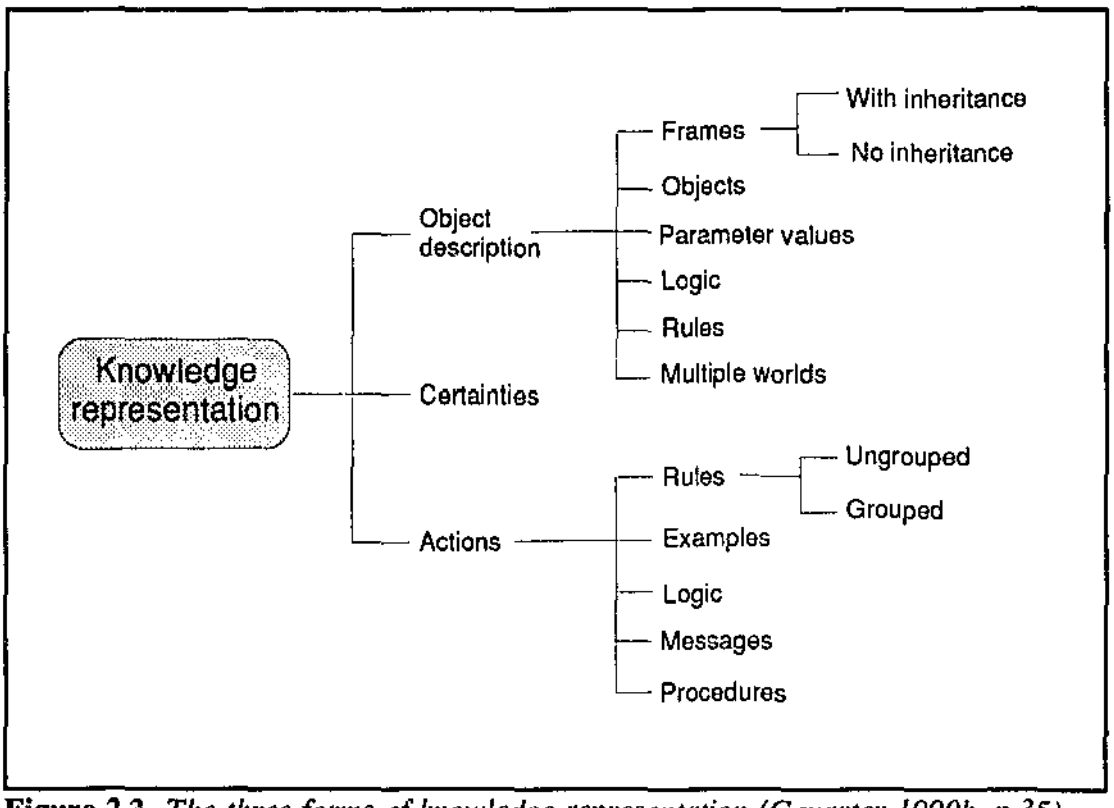

Figure 2.3 *The three forms of knowledge representation (Gevarter 1990b, p.35)*

One method often associated with expert system knowledge representation is *object description*. A common form of object representation is *Frames*, with or without the concept of *Inheritance*. Gevarter (1990b, p.35) states that: "inheritance allows knowledge bases to be organised as hierarchical collections of frames that inherit information from frames above them. Thus, an inheritance mechanism provides a form of inference". Frames

are tabular data structures of related slots for organising object representations (Gevarter, 1990b; Martin et al., 1988; Parsaye et al., 1988).

Another common method of knowledge representation is *Actions*. Though there are many forms of actions, they are commonly represented by *Rules*. Rules may be ungrouped or grouped into modules, the latter for easier maintenance and performance values (Gevarter 1990b, p.35). Rules are expressed in the general form (Parsaye et al. 1988, p.43):

If *condition*
Then *conclusion.*

A rival form in the actions group is *Examples*. Inductive systems benefit from them for knowledge acquisition. Gevarter (1990b, p.35) indicates that examples are a desirable form of representation as they are "much easier to elicit from experts than rules, and may often be a natural form of domain knowledge". Examples, unlike rules, are elementary pieces of knowledge.

Many expert systems have facilities for representing *certainty*, a measure to which the knowledge or data is correct. A common approach used for representing certainty, adopted in the Mycin expert system, is incorporating *confidence factors*. Others include *fuzzy logic* and *probability* (Gevarter, 1990b; Parsaye et al., 1988).

2.2.3 The Inference Engine

The inference engine is the central module of the expert system that contains the strategies for controlling the application of knowledge in the knowledge base (Martin et al., 1988, p.9). It is the reasoning mechanism of the expert system whereby the knowledge from the knowledge base is compared to the information supplied by the user and a deduction is made for relevant conclusions (Bielawski et al., 1988, p.28). There are several inference mechanisms depending on the type of expert system. The backward chaining and forward chaining mechanisms are common with rule-based systems.

Backward chaining. In backward chaining, the inference engine works by proceeding backwards from a hypothesis to those rules that have the hypothesis as an outcome (Gevarter, 1990b, p.35). An evaluation of this sort is done by *if-then* rules. Backward chaining is achieved by first finding a rule whose *THEN* part is the same as its hypothesis, then establishing the conditions contained in the *IF* part of the rule (Bielawski et al., 1988, p.29). This process is continued recursively until the hypothesis is fully supported or until a dead-end is reached (Gevarter, 1990a, p.21).

Forward chaining. The forward chaining mechanism is the inverse of backward chaining. This mechanism compares the information in the global database with the *IF* part of the rule. If the comparison between the global database and the *IF* statement reveals a match, the *THEN* part of the rule is added to the global database (Bielawski et al., 1988, p.32). Some systems incorporate *meta-rules* which determines the order in which the rules are evaluated (Gevarter, 1990b, p.36). Like backward chaining, the process of forward chaining continues recursively until a conclusion is reached or the

process becomes exhausted. For many problem domains, forward and backward chaining have been combined with interesting results (Gevarter, 1990b, p.37).

A number of tools offer a choice of several inference mechanisms which enable the developer to control the inference strategy (Gevarter, 1990b, p.37). Figure 2.4 shows the many inference types associated with expert systems.

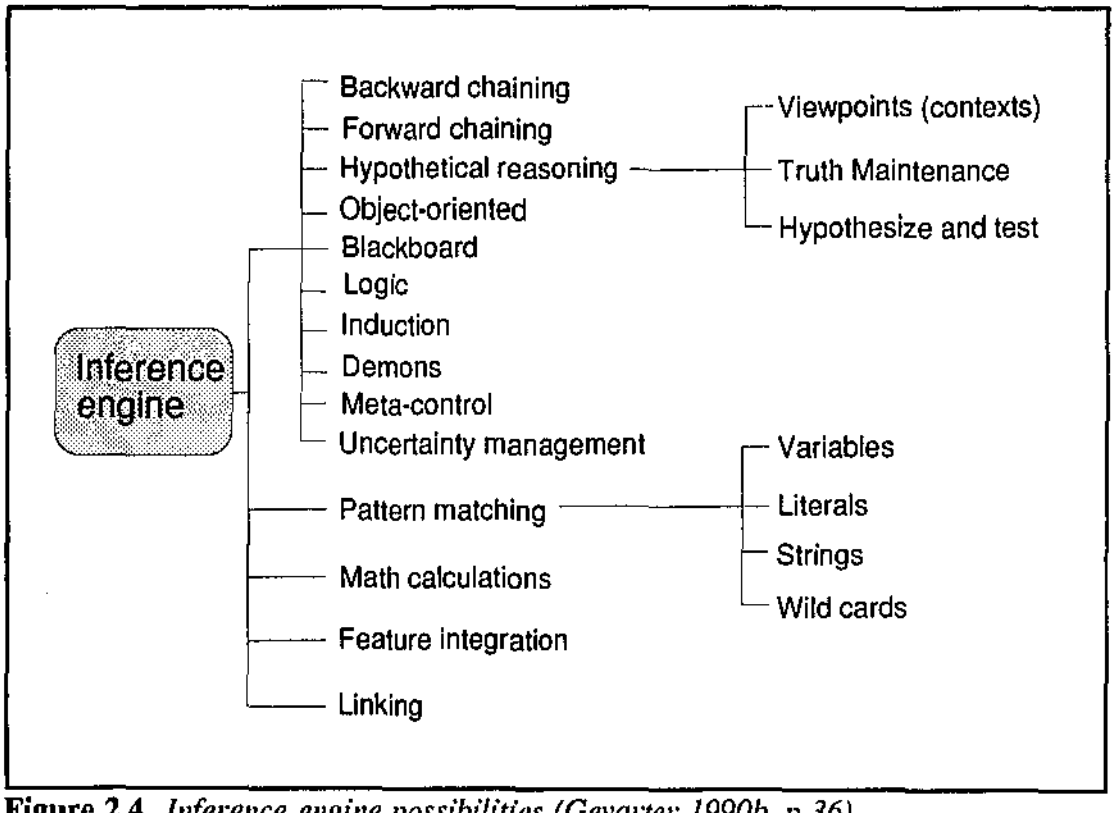

Figure 2.4 *Inference engine possibilities (Gevarter 1990b, p.36)*

2.2.4 User Interface

The user interface requirements for expert systems are categorised into those that support system developers and those that support the end users of the system, although Stelzner and Williams (1988, p.287) believe that the distinction between developers and end users is frequently obscured. This is due to expert advisory systems constantly evolving. Some tasks for example, which were considered knowledge acquisition are now being performed by end users.

User interfacing requirements for the two classes of users are not identical but in many instances do overlap. Interfaces designed for end users are useful for system developers, though not necessarily vice-versa as interfaces specifically designed for system development may be inappropriate to end users. The focus of interfacing for developers is domain representation and the reasoning process, as compared to end users in which the focus is on the domain itself (Stelzner et al., 1988, p.288).

Compared to traditional computer programs, Hendler and Lewis (1988, p.3) believe that expert systems have a different set of user interface requirements. Expert systems are not merely process implementing tools, but rather process *representations*. Often these processes coincide with conclusions that can result in crucial real world consequences. Thus, user interfaces for expert systems "must often present not only conclusions, but an explication of the processes by which those conclusions are reached" (Hendler et al., 1988, p.3).

Most expert systems are intelligent assistants in which the user interface is designed to allow interactive dialogue. Dialogues appear to users as structured data-input arrangements (many incorporating menu choices) that allow users to request for more information from the system based on a conclusive decision. In sophisticated systems, graphic representations of the reasoning process respond to the user's "how" questions. For simpler systems, a mere listing of the rules that support the system's conclusions may be employed to answer the user's queries. Rules are quoted for "why" questions. Gevarter (1990b, p.38) believes that the ability of the system to answer the user's "why" and "how" questions increases the user's faith towards the system's decision making process. Other dialogue facilities include "what if" queries which allow the user to select parameter values to observe the effect of an alternative decision outcome.

Many commercial expert system shells of today incorporate interfaces designed to support system developers (or knowledge engineers) whose tasks are to design and debug representations of the process while supporting, either through the same or a different interface, a user who requires an explanation of the decision process (Hendler et al., 1988, p.3). Sophisticated shells often incorporate interactive graphics and simulation facilities especially to enhance the user's understanding and control of the system (Gevarter, 1990b, p.38).

2.2.5 The Symbolic Languages

One of the important components in addition to the structure and paradigms supported by the expert system shell, is the programming language in which the shell was written in. The language used has a primary role in

determining whether the shell is compilable, and if so, whether compiled in a batch mode or incrementally. Compilable systems have the potential of reducing memory requirements and increasing performance speeds, while incremental compilability reduces development time (Harmon, Maus & Morrissey, 1988; Gevarter, 1990b).

Many sophisticated shells have been written in Lisp, as it deals with symbols, symbolic expressions and employs a unique *nesting* structural representation. This results in powerful problem solving techniques like searching (Harmon et al., 1988, p.35). However, many shells of today have been written in languages such as C to take advantage of increased speed, reduction in memory requirements and in a commercial sense, to support a wider variety of computers (Gevarter, 1990b, p.39).

Whether the shell was written in Lisp, Prolog or Pascal, these languages provide developers with the ability to *extend* the shell by writing additional functions. Similar extensibility found in modern shells is the integration of language *hooks* for interfacing with other programs that perform additional routines or database hooks for accessing other information. These attributes enable developers to design expert systems that are fully embeddable in other systems, enhancing system autonomy (Gevarter, 1990b, p.39).

2.3 THE COMMERCIAL USE OF EXPERT SYSTEMS IN THE USA

A survey questionnaire was conducted in 1990 in an effort to provide a comprehensive view of the commercial use of expert systems in the USA. The select group of 500 directors of information centres, data processing managers and systems analysts were members of the Association for Systems Management. To represent a fair sample of expert system users, the 500 different companies selected were of all types and sizes. (Ansari & Modarress 1990).

2.3.1 Findings of the Survey

Hardware. The expert system hardware of today was comparable to a conventional computer with the exception of a larger memory capacity and with speeds capable of running various AI software. Of the 175 usable surveys returned, Ansari et al. (p.11) states that:

> twenty-six of the total 42 companies surveyed indicated that the expert systems in use or under development use IBM PCs (XT, AT, or PS/2) and other compatible computers. The ever-increasing popularity of the PC in business, along with its growing efficiency and declining price, suggests that the PC will continue to increase its share of expert system applications.

Running a close second to personal computers was Symbolics, followed by the DEC VAX systems. The two main approaches for developing expert systems were through the use of AI languages (PROLOG or LISP), or through the use of commercial expert system shells. According

to the survey by Ansari et al. (1990), twenty-six percent of the companies surveyed reportedly used an AI language in the development of their expert system applications. Seventy-four percent of the companies surveyed used commercially available expert system shells.

Development Cost and Time. The size of expert system applications varied depending on their requirements. The largest reported application contained more than 4000 rules in its knowledge-base at a cost of more than US$1 million in a period of 36 months to completion. Conversely, the smallest contained 60 rules at a cost of between US$3000 to US$4000 in a period of two months to completion. According to the survey the average cost per rule, through the use of a commercial shell and a mainframe, amounted to US$810. Comparatively, using a commercial shell and a PC amounted to an average of US$115 per rule. (Ansari et al. 1990, p.12).

Benefits. The responding companies were then encouraged to share their benefits gained from the implementation of expert system applications. Regarded as the most important benefit by approximately 54% of the companies according to Ansari et al. (p.12), was the "improvement in the decision making of non-experts ... [Furthermore, a close] 45% agreed on consistency in decision making as the second important benefit ... response time in some decision areas is also faster, according to 28% of the companies".

The cost benefits agreed by twenty-one percent of the companies were the reduction of operational costs. This was due to the expert systems ability to perform some human tasks with considerably less cost. More than fourteen percent of the companies regarded expert systems as a great benefit in staff training. As stated by Ansari et al. (p.13), other benefits achieved through the

use of expert systems include "reduced business risk, improved products or service level, standardized communication, improved customer support, co-ordinated schedule planning, and reduction in paperwork".

Problems encountered. Of the problems encountered through the use of expert systems, 21.5% of the companies reported a lack of qualified knowledge engineers and expert system designers. More disturbing was the lack of commitment from top management, and a lack of domain experts - a recipe for project failure.

Other problems encountered include high development costs (19%); lack of compatibility with existing systems (19%); problems with the efficiency and accuracy of expert systems (16.7%) and the lack of facilities to support expert systems (12%). Ansari et al. (1990, p.13).

CHAPTER 3

REVIEW OF LITERATURE

Chapter Headings:

3.1 INTRODUCTION

Literature relevant to the research questions are reviewed in this chapter. Empirical findings in the literature are discussed and analysed to discover how they influence the work in this thesis. Topics covered include shell selection, knowledge representation and system interfacing.

3.2 CONCEPTS OF EXPERT SYSTEM SHELL SELECTION

3.2.1 Limitations of Shells

There are limitations associated with commercial expert system shells (or tools) available for small computers, including the limitations when evaluating them for potential development. Reichgelt and van Harmelen (1985) pointed out the shortcomings associated with two types of tools currently available for constructing expert systems[4]: the commercially available expert system shells, and high level programming language environments such as KEE and ART, which are also known as shells. The study concluded that different knowledge representation paradigms require different logics and control structures. Commercial shells and high level programming language environments fail to accomodate for different *models of rationality*. To appreciate these models of rationality, different tasks and

[4] The term *expert system* without the postscripts *shell* or *tool* (i.e. expert system shell) describes any system built with a knowledge-based approach.

different domains require different inference engines (Reichgelt et al., 1985, p.23).

Commercial expert system shells are usually constructed by abstraction from a working expert system. These shells include a built-in inference engine and an empty knowledge base with often primitive debugging and explanation facilities added to aid system developers. Contrary to what shell manufacturers often claim and the buyer's initial belief, these shells are not appropriate to many tasks and their respective domains. The study indicates that a common complaint amongst users is that the inference engines of commercial shells that have been successful in their first applications are not necessarily successful in the next. In addition, the knowledge representation of the shells are too rigid often making the expression of knowledge awkward (Reichgelt et al., 1985, p.21).

Many developers opt for high level programming languages which do not provide pre-fabricated inference engines. The programming facilities available enable knowledge engineers and system developers to design and construct inference engines to suit their applications. However, the down side associated with this capacity is that developers are bewildered with the endless possibilities and little guidance is given in undertaking them. Reichgelt et al. (1985, p.22) adds that "unless used by experienced programmers, high level programming environments encourage an ad hoc programming style in which no attention is paid to a principled analysis of the problem at hand to see which strategy is best suited for its solution".

Reichgelt et al. (1985) provides two sets of criteria which are relevant for adopting a control structure and a logic for a particular application. The

guidelines are not comprehensive and Reichgelt et al. recommended a more detailed paper to interested readers.

Buchanan et al. (1989) acknowledges that one of the problems facing developers when choosing a shell is the small upper limits on the size of the knowledge base. The reason being that many shells have been designed before powerful chips and large memories were available for the conventional computer. Even some modestly large and complex systems accomodate a few thousand rules. Buchanan et al. (1989, p.181) stressed that the rule limitations of today's large systems are mainly due to the system developer's (or knowledge engineer's) inability to keep track of the large number of items and interactions. Time restrictions imposed on projects also prevented them from building larger systems. There was no evidence to suggest that the rule limitations were owed to hardware or software limits. Buchanan et al. (1989) believes that with today's standards, to accommodate for large knowledge bases with millions of items, new technology will be required for managing knowledge efficiently.

Myers (1990, p.14) surveyed vendors and observed that many pre-1985 shells did not readily support the integration to another system such as database management systems. However, no further elaborations were made on this claim. Myers also points out that many shells that are Lisp- or Prolog-based are not portable from one environment to the next. A survey performed by Teknowledge (no date was declared) and cited in Myers suggests that many American and international firms are reluctant to transfer to expert systems due to their large investments in currently established hardware, software, operating systems and personnel training.

3.2.2 A Review of Shell Selection Criteria

Bielawski et al. (1988) provides a set of shell selection criteria that focuses on the end user's needs. The attributes covered lacked technicalities and are not comprehensive, hence they are more of an advice to the reader. Compared to other works, the selection criteria seems to cover a particular domain of a selection process. Bielawski et al. (1988, p.87) provides five attributes in the selection criteria which includes:

a) Fit of the tool to the problem.

b) Effectiveness of the developer interface.

c) Effectiveness and friendliness of the user interface.

d) Integration capability with existing programs and databases.

e) Run-time licensing for delivered systems.

Bielawski et al. (1988) covers seven products in the rule-based, induction and hybrid category of shells and discussed them in detail based on the five attributes. Although their work offers a comprehensive insight into the seven shells, the style adopted by Bielawski et al. restricted further attempts to analyse other shells. The seven shells reviewed are representative of their respective categories.

Bowerman et al. (1988) also provides a set of criteria known as *strategies* for expert system shell selection. Like Bielawski et al. (1988), the selection criteria focuses on the needs of the end user. The strategies are advice on what to look for in a shell, but did not detail on aspects as *how* to look for the attributes in a shell. However, another section of Bowerman et al. provides several tables listing thirty current shells on the market and their respective features as a comparison. Additional attributes listed that are not covered in Bielawski et al. (1988) include machine requirements of the

shells, rule capacity, response times, pricing and availability, and vendor stability and growth. Bowerman et al. (1988, p.96) advises that shells should be carefully selected to match the different phases of the application to:

a) increase the chances of success,

b) speed development,

c) improve the quality of the finished product and

d) to save money.

Citrenbaum, Geissman and Schultz (1990) provides an interesting comparison of four types of expert system users: the student, domain expert, knowledge engineer and the expert system software developer. The paper details the important features of expert system shells to consider in relation to the requirements of these users.

Gevarter (1990b, p.50) did not attempt to produce a selection criteria, but provides considerations for assessing the overall usability of a tool (see Figure 3.1) which generally, summarises the selection criteria listed by Bowerman et al. (1988) and Bielawski et al. (1988).

Martin et al. (1988) provides a different set of selection criteria that covers technical and business issues related to the purchase and use of expert system shells. The criteria forms part of a larger shell selection paradigm which includes a comprehensive shell selection methodology (see 4.3). Sample score-sheets are also provided to help the reader in evaluating shells. Unlike the works of Bowerman et al. (1988) and Bielawski et al. (1988) which details shell features to consider when selecting, Martin et al. (1988) caters for a possible methodology for shell selection.

Rothenberg (1989) presents a framework of evaluation criteria and a methodology for selecting an expert system shell for a given task. The issues presented are more detailed and technical than the works of Martin et al. (1988). The framework also reveals the strengths and weaknesses of individual shells.

Brownstein and Lerner (1982) gives a comprehensive methodology for the formalities on the selection of software packages. Although designed to cover the selection process of all types of software packages, the methodology can be adopted and combined with the works of Martin et al. (1988) and Rothenberg (1989) to produce an adaptable expert system shell selection methodology.

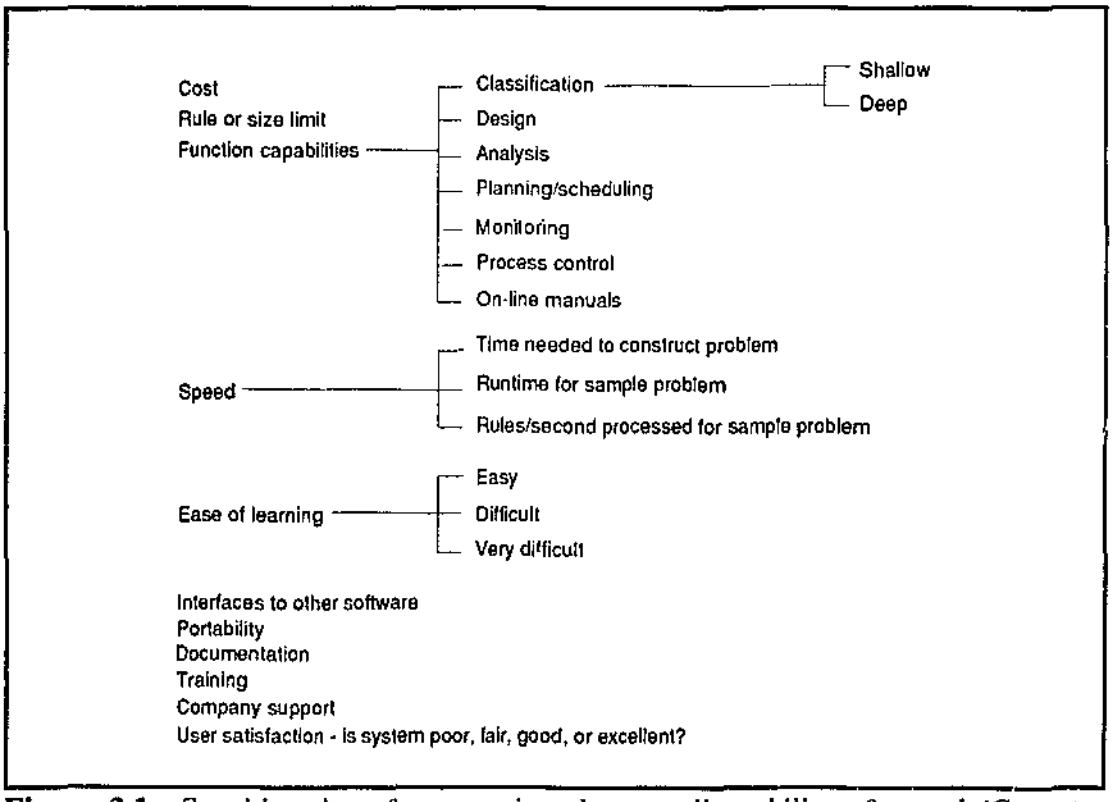

Figure 3.1 *Considerations for assessing the overall usability of a tool (Gevarter 1990b, p.50)*

3.2.3 A Review of Shell Comparisons

Gevarter (1990b) and Freedman (1990) identified the features of shells that are important for solving certain types of problems. Twenty shells were compared in Gevarter while Freedman compared 27 with respect to features and problem types. Harmon et al. (1985) organised shells into classes, evaluated seventeen shells with respect to knowledge representation, inference and control strategies, and identified the best shells for particular problem types. These works provide a comprehensive comparison of popular, well established shells on the market which can be used to short-list potential shells for the PESWEA project. The comparisons are listed in a tabular format which exhibits the shells' strengths in relation to the listed features. However, these comparisons mean that new shells are not evaluated and compared to established shells.

Barr (1990) reviews three shells circa 1988: ESIE, EXSYS v3.0 and VP-Expert v1.2 based on what they have to offer and the presentation of their respective features. Raeth (1990) reviews two shells: Clips v4.11 and Personal Consultant Plus v3.02 based on their capabilities to implement a Statistical Strategist application. These shells were new at the time the papers were written in 1988, and may already be well established in the marketplace. These reviews provide much needed details for future shell comparisons.

3.3 KNOWLEDGE ACQUISITION AND REPRESENTATION

3.3.1 Knowledge Engineering

Knowledge engineering as defined by McGraw and Harbison-Briggs (1989, p.5) is "the term used to describe the overall process of developing an expert system". Building an expert system involves information gathering, domain familiarisation, analysis, and design efforts. The knowledge accumulated in this process must be translated into code, tested, and refined. Therefore, the goal of knowledge engineering process is (McGraw et al. 1989, p.5):

> to capture and incorporate a domain expert's fundamental domain knowledge, as well as his or her prediction and control processes. The end result of the knowledge engineering process should be strong, robust performance on the part of the expert system.

The process of knowledge acquisition does not occur as a single step in the system development, rather it has a role in every step of the development process. This is so, as knowledge acquisition is an ever-changing process.

3.3.2 Identifying the Sources of Expertise for Knowledge Acquisition

The process of knowledge engineering cannot begin until the source of expertise is identified. This requires initial preparatory work, and Hart (1989, p.50) has identified several aspects:

Understanding the problem. The problem domain may have been defined beforehand, however the person involved in the knowledge engineering process (known as the knowledge engineer) will still have to understand the problem. Understanding the problem may involve observing the daily operations of the staff by talking to people or monitoring departments. By analysing the workings, the knowledge engineer will then know how things fit into the picture.

Written material. Departmental manuals and necessary documents relating to the subject area should be identified and studied. Written materials are reviewed by the knowledge engineer to determine what information each document contains and whether they can be used without assistance. Scott, Clayton & Gibson (1991) believe that if the knowledge engineer can obtain information from documents with or without assistance, the total time required for knowledge acquisition can be significantly reduced. Hart (1989) argues that documentation often does not describe the expert's[5] knowledge, but acknowledges that written documentation does provide a good source of reference for the subject area.

---

[5] *Expert* as defined by Scott et al. (1991, p.481) is a person whose knowledge and experience in some particular field exceed the average and whose performance of tasks related to this field is above average.

Identifying the experts. For simple projects, usually one expert is sufficient to provide both the knowledge and available time for the knowledge acquisition process. The expert also has the authority to judge the accuracy of the expert system's behaviour.

3.3.3 Knowledge Acquisition Methodology - A Review

Breuker and Wielinga (1987, p.20) believe that the current practice in knowledge acquisition is not systematic and is often characterised by *rapid prototyping*. Rapid prototyping (McGraw et al. 1989, p.347) is the selection and rapid development of a section of the expert system, testing on the partial system, iteractive refinement, and further development. Hayes-Roth et al. (cited in Breuker et al. 1987, p.20) claims that the "process of building expert systems is inherently experimental". Thus, the process of knowledge acquisition has at least some degree of trial and error, and according to Breuker et al. (1987), has discouraged the development of a standard methodology. Indeed, the works of Hart (1989), McGraw et al. (1989) and Scott et al. (1991) did not provide a standard and concise knowledge acquisition methodology, but rather a set of guidelines to adopt.

Several knowledge acquisition methodology tools have emerged through field research and experimentation which includes KADS, ROGET and PKA (Breuker et al. 1987; Naughton 1989). Empirical evidence have shown that the methodologies are effective in development and superior to rapid prototyping, but are not as yet conclusive for several reasons. This includes the adoption of the methodology for new projects that have yet been made operational (Breuker et al. 1987, p.40).

3.3.4 Systems Analysis and Knowledge Acquisition - A Comparison

Hart (1989), and McGraw et al. (1989) extended the role of systems analysis into the process of knowledge acquisition in expert systems development. Systems analysis has evolved over a number of years in the field of application development and is the process whereby a system is evaluated and analysed, often with a view to computerising some or all of it (Hart 1989, p.32). Hart and McGraw et al. acknowledged that many formal methodologies on systems analysis have been developed and there is no shortage of literature on the procedures involved, however many projects continue to fail as program errors are located and fixed one by one. Failed projects can be attributed to the difficulties of project management with some problems highlighted by Hart (1989, p.35) as being:

- failure to agree objectives,
- failure to ask questions,
- failure to be specific,
- forgetting answers,
- ignoring suggestions,
- making false assumptions,
- failure to explain the consequences of a decision,
- and misunderstanding of terms/jargon.

As expert systems are essentially programs, Hart (1989) believed that the process of knowledge acquisition is comparable to systems analysis. As a brief overview, the main stages of systems analysis are highlighted as follows (Hart 1989, p.32):

a) *Project selection.* An appropriate application is selected for implementation.

b) *Feasibility study.* Provides cost-benefit analysis of alternative approaches to project and defines the requirements of the system.

c) *Analysis.* The analysis of the system follows the results of the feasibility study, providing a basis for the design.

d) *Design.* Details what is to be done to the system and how they are to be achieved.

e) *Development and testing.* Programs and documentation are written, and program undergoes testing for conformation to specifications.

f) *Changeover and use.* The old system is changed over to the completed new system.

Hart (1989, p.39) believed that the stages of systems analysis is much simpler compared to knowledge acquisition. The process of systems analysis evolves around definitive information. The systems analyst has a fair idea of what is required and questions usually involve how, why, when, how many and the like.

However as Hart (1989, p.39) pointed out, this is not the case with knowledge acquisition. Domain experts hold a lot of knowledge accumulated through understanding of their task, intuition, and skill which are not defined as a set of procedural rules. As the routine of knowledge acquisition is not well defined, a whole new set of difficulties arise. The subdivided activities of knowledge acquisition throughout an expert system development can be defined as the following tasks (McGraw et al. 1989, p.12):

- Initially entering knowledge
- Reducing or avoiding erroneous knowledge
- Augmenting acquired knowledge

This view is presented in a framework for knowledge acquisition in Figure 3.2. The major stages are identification, conceptualisation, formalisation, implementation and testing:

Identification. Tasks are selected for suitability to expert systems. Objectives are defined and knowledge engineer becomes familiar with the domain and facilities. (Hart 1989; McGraw et al. 1989).

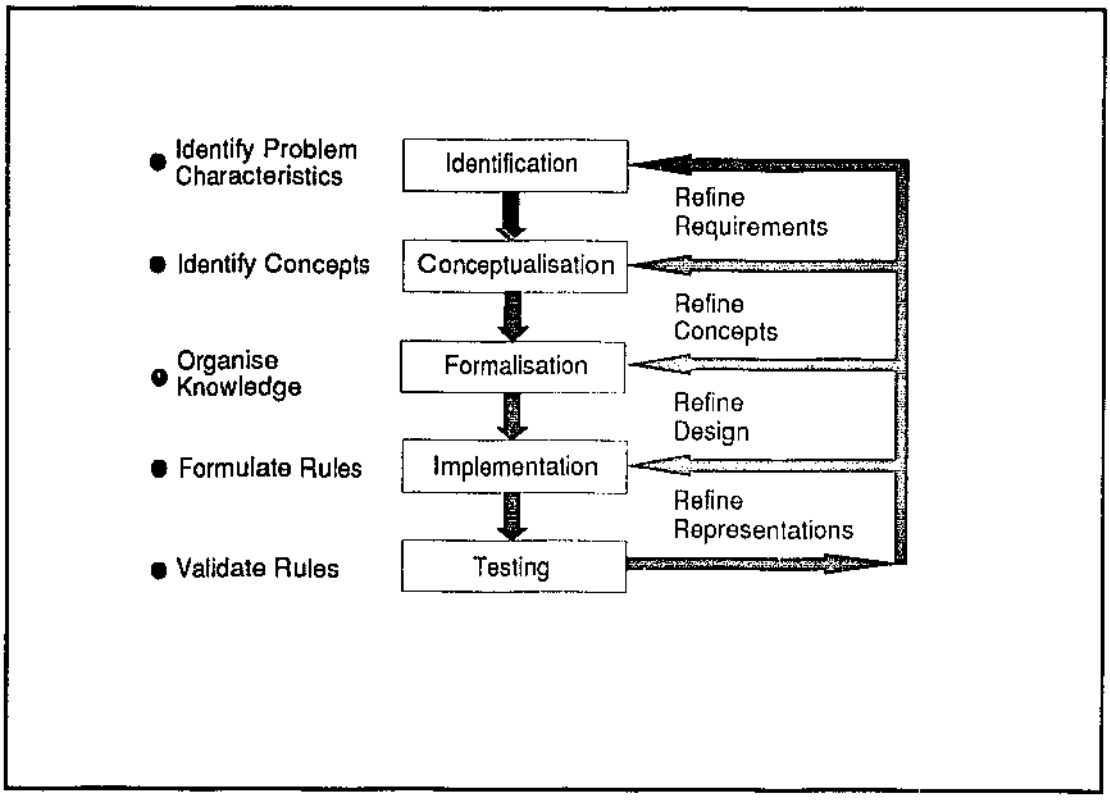

Figure 3.2 *A representative knowledge acquisition framework used in the expert systems industry. (McGraw et al. 1989, p.12)*

Conceptualisation. The knowledge is extracted and represented to form a conceptual model which depended on the domain being studied. The conceptualisation of the problem required consultation with experts and users. (Hart 1989; McGraw et al. 1989).

Formalisation. This design stage involves selecting appropriate structures to represent the knowledge base and inference mechanism in the expert system. (Hart 1989; McGraw et al. 1989).

Implementation. The design details are then implemented into the system which involves extracting knowledge into representational frameworks for the selected expert system shell. (Hart 1989; McGraw et al. 1989).

Testing. The system is tested for accuracy, adherence to design and specifications, and knowledge acquisition efficiency. A test scenario is selected for testing in which the results are used to revise the prototype. (Hart 1989; McGraw et al. 1989).

3.3.5 Analysing and Making Conclusions

Scott et al. (1991, p.228) described several representations that are used to organise and illustrate *judgmental* knowledge. This activity is the analysis of the expert's knowledge, which Scott et al. defined; "specifies how the expert system can use currently known facts and currently believed hypotheses about the case to conclude new facts and hypotheses" (p.228).

General Decision Tables. An inference[6] is made up of a *condition* and a *conclusion.* A condition is also known as a *basis characteristic*, while the conclusion is known as the *conclusion characteristic.* Generally, Scott et al. (1991, p.229) perceived inferences as having the form:

If *basis characteristic 1* = *value 1* and
basis characteristic 2 = *value 2* and
...
basis characteristic N = *value N*
then *conclusion characteristic* = *value M*

Basic Characteristic 1	Basic Characteristic 2	...	Basic Characteristic N	Conclusion Characteristic
value 1.1	value 2.1	...	value N.1	conclusion A
			value N.2	conclusion B
			...	...
			value N.R	conclusion C
	...	...	...	...
	value 2.Q	...	value N.1	conclusion D
			value N.2	conclusion E
			...	...
			value N.R	conclusion F
...	...	...	...	...
value 1.P	value 2.1	...	value N.1	conclusion G
			value N.2	conclusion H
			...	...
			value N.R	conclusion I
	...	...	...	..
	value 2.Q	...	value N.1	conclusion J
			value N.2	conclusion K
			...	...
			value N.R	conclusion L

Note: Letters A through R represents a small integer.

Figure 3.3 *A general decision table (Scott et al., 1991, p.231).*

[6] A decision that adds to an expert's body of knowledge about a case (Scott et al. 1989, p.160).

Decision tables serve to capture the above standard inference structure. The decision table is a two-dimensional matrix structure that indicates what conclusion characteristics can be deduced from all of the possible combination of basis characteristics (Scott et al. 1991, p.229). Two main types of decision tables exist of which the first is the *general* decision table (to be discussed here), and the second is the *two-basis* decision table. Hart (1989) and Scott et al. (1991) acknowledged that decision tables are invaluable to knowledge engineers who use expert system shells of the induction type (see 3.3.6). Figure 3.3 illustrates a general decision table.

The columns in the table corresponds to a basis characteristic, while the right-most column coincide with the conclusion characteristic. A cell or row, specifies a value for the appropriate characteristic. The decision table can show at a glance, how the expert system should be able to deduce a conclusion characteristic from the spectrum of circumstances that may be encountered by the system.

Pseudorules. Decision tables however, suffer from several limitations as outlined by Scott et al. (1991, p.233):

- They are not appropriate for expressing *complex conditions* on the basis characteristics.
- They are not appropriate if *different inferences use different collections of basis characteristics.*
- They are not appropriate if the *form of the condition on the basis characteristics varies* in different circumstances.

Decision tables are inadequate for representing heuristics. Pseudorules are able to record heuristics often used by experts to make a specific type of inference (Scott et al. 1991, p.234). Rule-based expert systems capitalise on pseudorules. As illustrated in section 2.2.2, pseudorules take the form of:

If *condition*
then *conclusion*

Decision Trees. A decision tree is a type of flow chart consisting of nodes and branches which can illustrate the process by which the expert reasons. Nodes of the tree correspond to a question, while the branches of that particular node coincide with the answers. A path of the tree represents a situation to which the expert system may encounter. The conclusion to the path is represented by a leaf node. Decision trees are able to illustrate the sequence of steps that the expert system should take to infer a conclusion characteristic (Hart 1989; Scott et al. 1991). Figure 3.4 illustrates a typical decision tree.

3.3.6 Inductive Systems for Knowledge Acquisition

With inductive systems, human experts can supply examples of problems and their associated solutions into the expert system shell, and by selecting a command the shell then automatically generates the appropriate rules from the given examples (Coleman 1989, p.174). Figure 3.5 illustrates the principles of inductive systems (Hart 1989, p.124). An example of an inductive expert system shell is *1st-Class*. Although simple and useful, there have been some controversy over the role of inductive systems in the field as many critics have dismissed inductive systems as useless (Coleman 1989; Hart 1989).

The main problem with inductive systems is that during development, every possibility of a particular problem domain will have to be identified and entered into the system as examples. The knowledge engineer will have to be careful to enter the correct set of examples as the laws of *GIGO* (Garbage In Garbage Out) will apply (Coleman 1989, p.176).

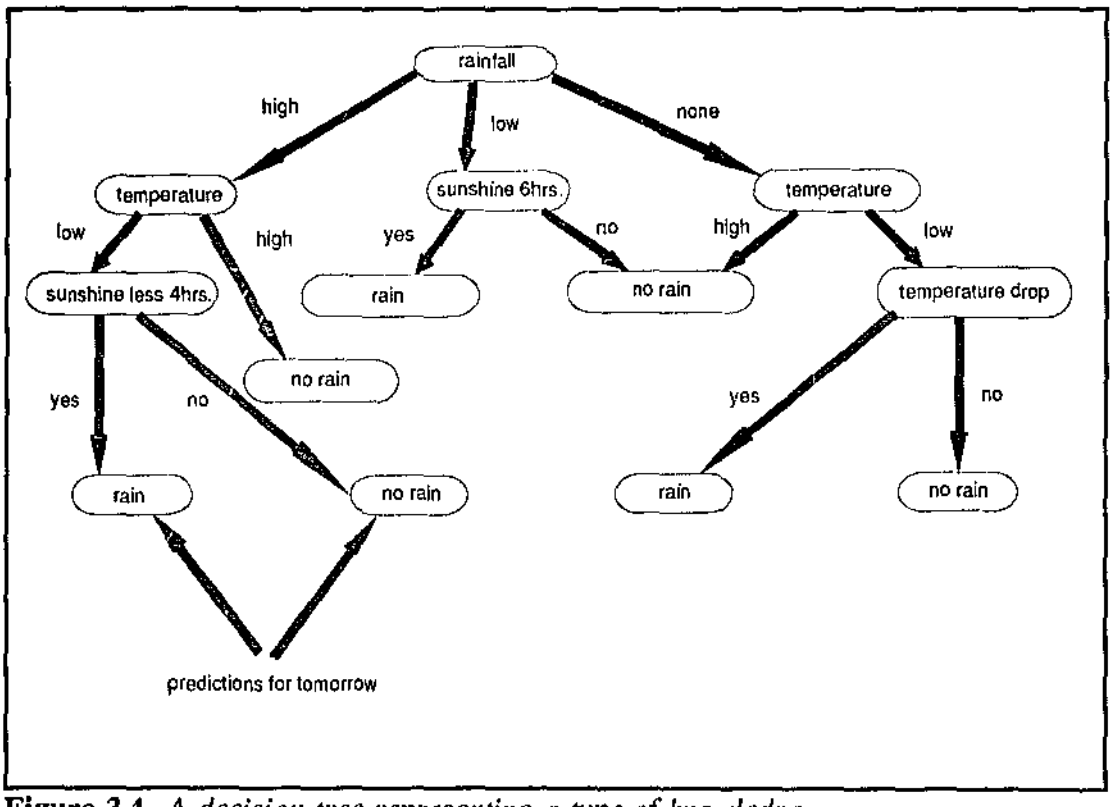

Figure 3.4 *A decision tree representing a type of knowledge.*

The arguments for inductive systems include easier and faster knowledge acquisition as knowledge recalled by human experts can immediately be represented as examples. Gevarter (1990b, p.35) states that "examples are much easier to elicit from [human] experts than rules, and may often be a natural form of domain knowledge". Coleman (1989) and Hart (1989) believe that induction systems are only useful for some types of problems, particularly for smaller systems and for prototyping during the earlier stages of program development.

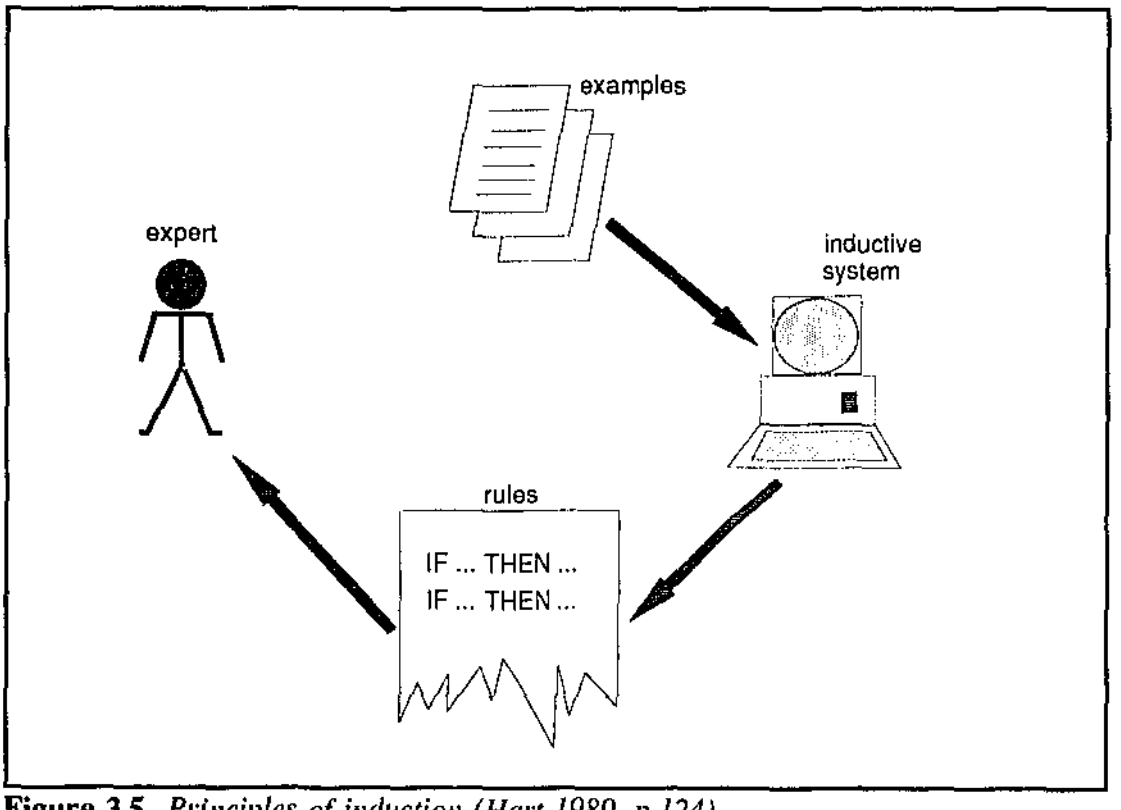

Figure 3.5 *Principles of induction (Hart 1989, p.124).*

3.3.7 Common Knowledge Representation Types

According to Ramsey and Schultz (1989, p.273), most of the knowledge representation paradigms in use for current expert systems "were developed in the mid-1960s to mid-1970s as a result of a debate over the merits of procedural knowledge versus declarative knowledge". The use for each knowledge representation paradigm is domain dependent, however as Ramsey et al. (1989, p.274) found, there are a set of features to consider when choosing a representation paradigm appropriate for the application:

Ease of domain representation. The application domain should have a major influence on the particular knowledge representation paradigm. Among the things to consider include the ability of the paradigm to describe every aspect of the domain in a concise way; consideration of the existing application knowledge format; and the intended use of the expert system.

Representation efficiency. The representation paradigm must be able to express the knowledge in the least amount of space required without being time consuming.

Ease of understanding. The human expert must be able to read and understand the knowledge base to evaluate for completeness and accuracy of the information; for maintenance purposes, and for further knowledge acquisition.

Uncertainty. As many real-world problems often have uncertainty factors, the knowledge representation paradigm must be able to accommodate reasoning with uncertainty.

3.3.7.1 Rule-based representation.

Used for rule-based expert systems which consists of rules, a rule interpreter and a working memory. The knowledge is represented as a set of rules (see 3.3.5) in the form of condition-action pairs. The action of a rule is triggered by the rule interpreter when a condition is satisfied by elements contained in the working memory (Ramsey et al. 1989; Parsaye et al. 1988).

Advantages and Disadvantages. Rule-based representations are appropriate for representing knowledge in independent modules of information which are easy to understand. Rule-based representations have been successfully applied in the domains of design, diagnosis, interpretation, monitoring, and planning. Pattern-action statements can be easily expressed with rule-based representation, but it does not allow for natural representation of highly structured information (Ramsey et al. 1989, p.280).

3.3.7.2 Semantic networks

Semantic networks are used to represent hierarchical information. A language that takes advantage of semantic networks is Prolog. Pictorially, semantic networks are nodes which represent classes or instances of objects, and links which represent the relationships and characteristics of the classes or instances of objects (Ramsey et al. 1989; Merritt 1989). Semantic networks are a collection of binary predicates such as:

ISA(Tom, elephant)

ISA(elephant, mammal)

Body-covering(mammal, hair)

Advantages and Disadvantages. Searching for information is easier as related facts are linked next to each other. Hierarchies can be represented due to the connections between classes and objects. However, there are no formal

ways of dealing with semantic networks, and structures can become complex as the application grows making searching inefficient (Ramsey et al. 1989, p.285).

3.3.7.3 Frames

Frame-based knowledge representation combines both data and procedures into a structure known as a frame. Organised into hierarchies, frames can be used to inherit information from *parent* frames. Frames are composed of record structures known as *slots* which contain values of a particular object, concept or event. The structure of a frame consists of (Parsaye et al. 1988, p.163):

The name of the frame.

The parent of the frame.

The slots of the frame and the associated values.

Attached predicates for each slot.

The following is an example of a frame:

Frame: John
Parent: Person
Salary: $100,000
Want_to_buy: House-1 *(name of another frame)*
Bank: The_Which_Bank-1 *(name of another frame)*
Afford: Calculation-1 *(name of procedure)*

Procedure: Calculation-1
Down_payment: 0.1 * House-cost
Loan_interest: 0.5
If(Salary_of_person * 4) *is greater than* House-cost
Then return OK
Else return Not_approved

Advantages and Disadvantages. Frames are used in large, advanced expert system shells like KEE and ART. The structure of frames provides immediate access to information which allows for efficient searching. Hierarchies of information can be represented in modules which provides easier maintenance and modification of frames. However, like semantic networks, Ramsey et al. (1989, p.289) believes that there are no formal methods of dealing with frames.

3.4 COUPLING EXPERT SYSTEMS AND DATABASE SYSTEMS

One of the most common commercial applications running on computers today is the database management system (DBMS). Many authors believe that the combined use of DBMS and expert systems is potentially very valuable for modern business applications. By enhancing DBMS with expert system features, it can be used more intelligently and may operate more efficiently. Furthermore, due to the widespread use of DBMS, the operational data required by the expert system can be readily made available from online databases (Al-Zobaidie & Grimson 1987; Jarke & Vassiliou 1984; Gillies 1991). This section will focus on how expert systems and DBMS can serve and collaborate with each other to provide a more powerful system.

3.4.1 The ES-DB Integration

Through integration, the benefits of conventional software engineering can be gained along with the special features of expert systems. Torsun and Ng (cited in Gillies 1991, p.185) listed a number of advantages which may emerge from the integration:

- Expansion of the power and flexibility of a DBMS by the inclusion of an inference mechanism.
- Provision of complex operations for data manipulation, beyond the scope of current DBMS.
- Acceptability by users who are unfamiliar with programming concepts.

- Provision of a large number of facts for the expert system.
- Provision of multi-user facilities for expert systems.

However, there are inevitable disadvantages associated with integrating expert systems with DBMS. Gillies (1991, p.62) presented a few which may arise from the integration:

Extra complexity. There are design issues to be considered when integrating expert and database systems. The interfacing may be more complex due to the inconsistent structures and coding styles between the two systems.

Finding a suitable design approach. Different methodologies and tools have been produced to support the development of each system due to their respective characteristics. However, no methodologies and tools exists for the integration of both. Creativity on the developer's part is required for a successful solution.

Unknown benefits. If expert systems technology is still in the process of acceptance in the business field then the concept of integration between the two systems is virtually charting the unknown. Although integrated systems already exists in the field, their potentials are endless and their benefits are still being assessed.

3.4.2 Classes of Interaction

Expert systems and database systems can interact in a variety of ways. The benefits to be gained with each approach depends upon the types of applications in which they are to be used. Al-Zobaidie et al. (1987) examined

the approaches adopted by existing systems and have divided them into three classes:

i) The intelligent database approach.

ii) The enhanced expert system approach.

iii) The inter-system communication approach.

The interested reader is referred to Jarke & Vassiliou (1984) for a comprehensive coverage of ES-DB interaction.

3.4.2.1 The intelligent DB

The intelligent database (DB) is produced by embedding a set of routines (or a deductive component) into the database management system, thus enhancing the efficiency and functionality of the DBMS. Artificial intelligence techniques like query optimisation helps to improve efficiency. The functionality of a conventional DBMS can be enhanced through the support of Natural Language query interfaces, multiple views, and the provision of mechanisms to handle incomplete data (Al-Zobaidie et al. 1987, p.30). Figure 3.6 illustrates three possibilities of incorporating the deductive component into DBMS. In model (a), the Integrated Method, the component constitutes part of the DBMS. Model (b) presents the Deductive Filtering method which filters user and subsystem queries through the deductive routines before being received by the DBMS. In model (c), known as the Interactive Method, the DBMS interacts with the user or the subsystem.

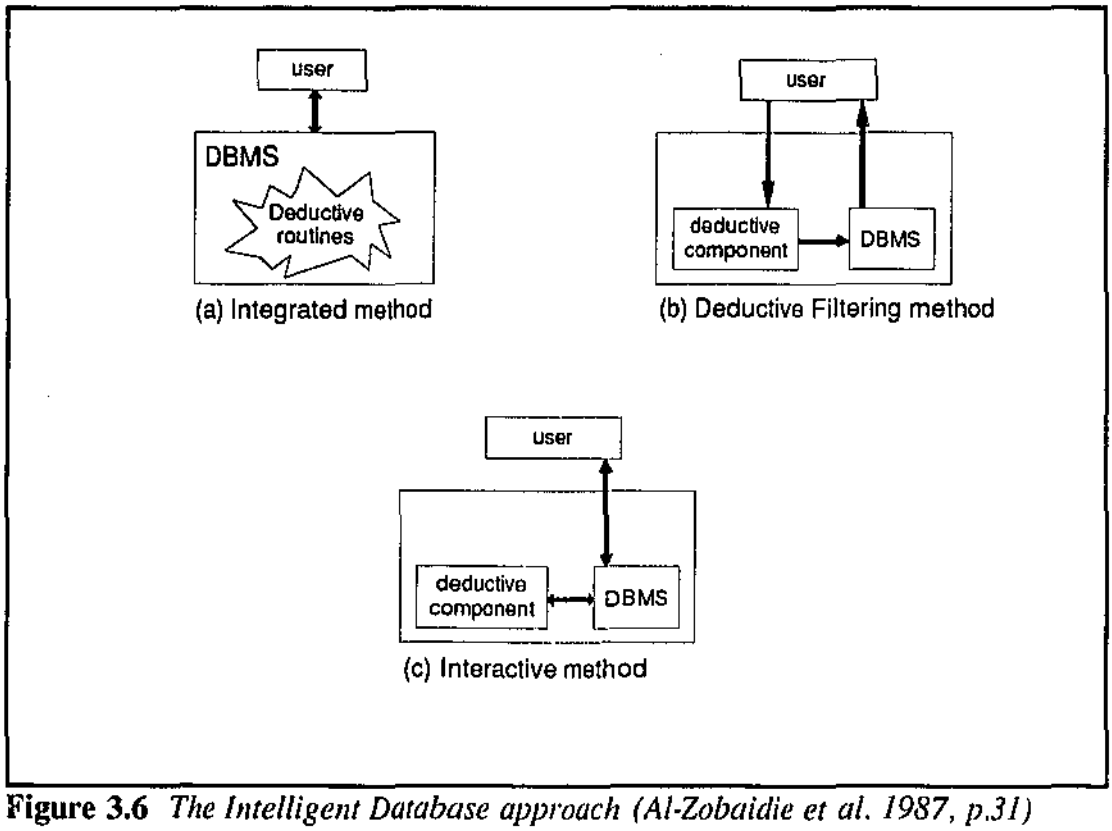

Figure 3.6 *The Intelligent Database approach (Al-Zobaidie et al. 1987, p.31)*

3.4.2.2 The Enhanced ES

While the intelligent database approach serves to enhance the power of the DBMS, the enhanced ES approach ultimately enhances the power of the expert system by incorporating extended data management facilities. Al-Zobaidie et al. (1987, p.30) dicussed two ways of enhancing the ES, as Figure 3.7 illustrates. In model (a), the Language Extension method, the programming language in which the expert system has been written in, is extended for system enhancement. This approach may be more feasible for expert systems designed from the ground up with a particular programming language like Prolog.

Model (b) depicts a more general approach where no modifications have been made to the existing languages. The expert system is enhanced by providing its inference engine direct access to a DBMS. Again there are two ways to achieving this approach, the first of which is the loose coupling of the ES with an existing DBMS (Al-Zobaidie et al. 1987; Jarke et al. 1984). Loose coupling does not permit a dynamic link between the two systems. Instead, communication is handled by downloading data as *snapshots* from the DB to the ES prior to operation. Alternatively, tight coupling permits a dynamic link, in which case the data can be retrieved from the DB when it is needed during system run-time.

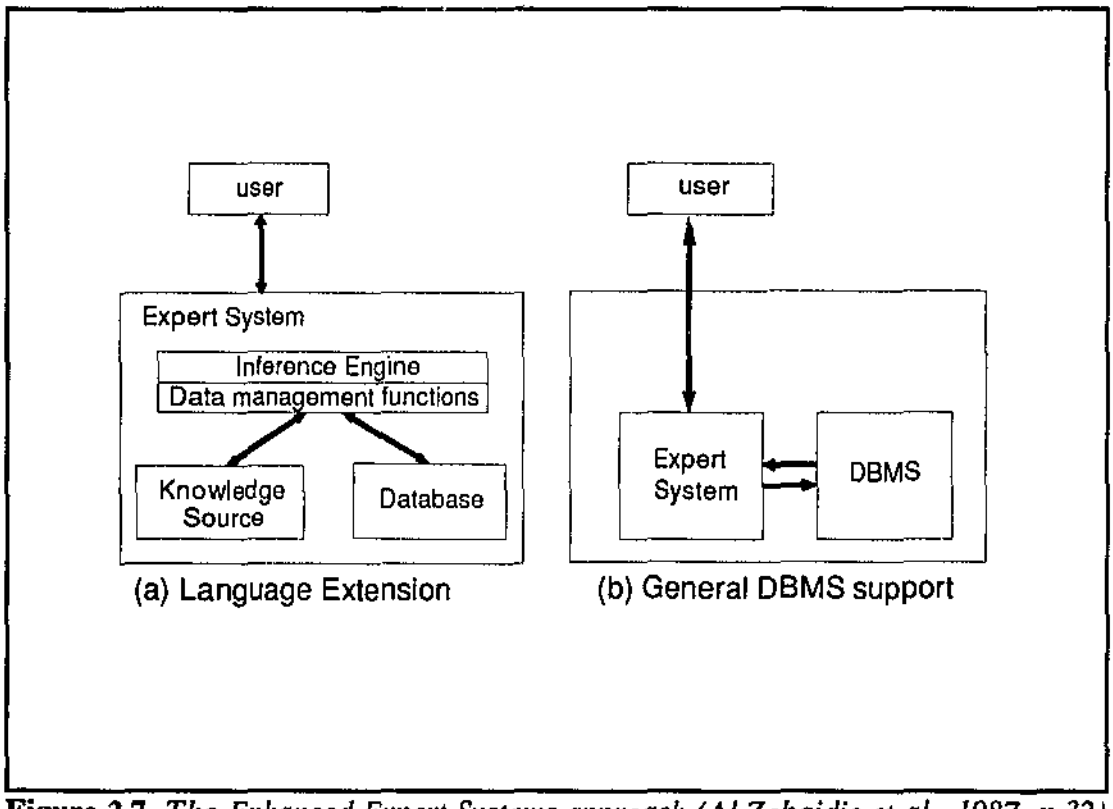

Figure 3.7 *The Enhanced Expert Systems approach (Al-Zobaidie et al., 1987, p.32)*

3.4.2.3 Inter-system communication

In this approach, the ES and DBMS co-exist as independent systems while communicating with each other. This permits either system, e.g. DBMS, to operate individually with its own set of users. In adopting this approach, planning is required to decide on where the system control lies; particularly the location of processing and interaction control. Al-Zobaidie et al. (1987, p.31) discussed three possibilities as illustrated in Figure 3.8.

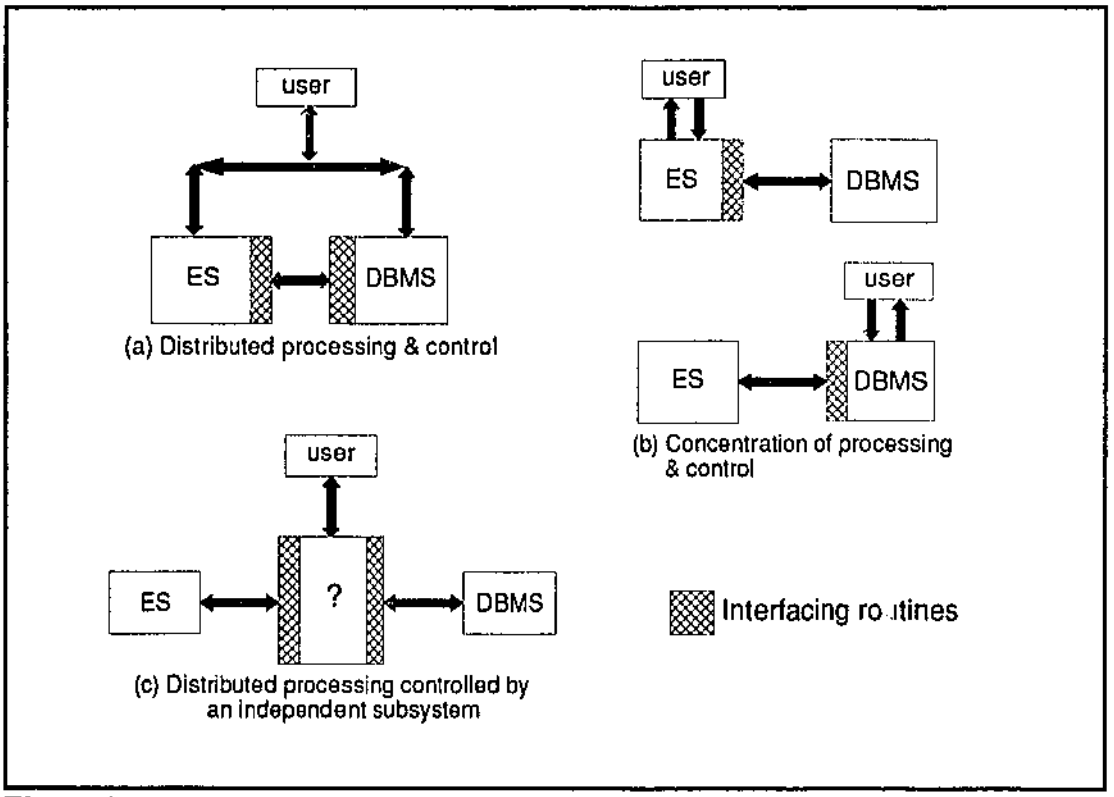

Figure 3.8 *The Inter-system Communication approach (Al-Zobaidie et al. 1987, p.32).*

Model (a) depicts the distributed processing and control method. Either the ES or the DBMS can operate independently and interaction is performed by message passing. As this approach permits system independence, it is possible to build upon the system by incorporating other expert systems and database management systems. However, such power and flexibility lures potential drawbacks including inconsistencies, incompatibility and redundancy (Al-Zobaidie et al. 1987, p.31). Model (b), depicts the concentration of processing and control method in which either the ES or the DBMS can dominate. While providing more flexibility than the first, Al-Zobaidie et al. (1987) believes that difficulties can still arise from the integration. Model (c) provides an alternative solution to the first two. An independent sub-system controls the interaction between the two systems, and the processing is now more distributed.

3.5 A CASE STUDY: SECV'S EESI PILOT SYSTEM

The following case study is a concise account of the State Electricity Commission of Victoria's (SECV) successful knowledge-based Employee Entitlements Source Input (EESI) pilot project. The EESI pilot system has relevance to this study as it was an expert system designed to replace manual timesheet processing. The system incorporated an award inference engine formulated to process and validate employee data. The successful outcome of the project during trial-runs as recent as 1989 prompted critics to acclaim it as being unique in the world. A potential savings of an estimated 2.8 million dollars a year have been predicted through the use of the completed system (Plant et al., 1990). As a result, the EESI system was cast as a role-model for many similar award-related studies rapidly emerging from the industry.

3.5.1 The EESI System

IBM assisted with the EESI system development which was implemented with IBM's Expert System Environment (ESE) shell. The host computer accomodated ESE under TSO. ESE/PC running under DOS on IBM PS/2 workstations accessed data on OS/2 databases through an IBM token ring network. Systems interfacing were made possible with routines coded in the C language. ESE facilities were extensively employed in the system for user interfacing. Further analysis suggested that the development team was satisfied with ESE's ability to adapt entirely to their requirements as there were extensive equipment re-tooling to accommodate the shell.

3.5.2 EESI Development Methodology

The EESI pilot development process was defined in five distinct stages:

(1) The search for an appropriate technology. Over a two month period the project team searched for an appropriate technology most adaptable to the business requirements of their EESI project.

(2) Vendor demonstration. A demonstration knowledge-base specific to SECV's payroll problem domain was built by IBM and demonstrated over two days to substantiate whether expert systems, and in particular IBM's own ESE shell, could be utilised to meet the application requirements.

(3) Prototype development. Over a three week period, a prototype was developed to further justify the expert system's appropriateness to the problem domain, and to identify system functional specifications. In addition, the prototype study was used as a form of training into knowledge engineering for the development team.

(4) Pilot development. Three independent development parties were involved in the pilot implementation over a period of five months. The main tasks of knowledge engineering and the delivery of the system were commissioned to IBM. Critical to the success of the pilot system was user acceptance. Although to be used by a distinct group, the system was assessed by a spectrum of users ranging from the unions to the end-users.

(5) Production. Following the success of the pilot system through extensive testing, SECV proceeded to develop the final production system in mid-1990 in a bid to utilise an estimated annual savings of 2.8 million dollars and an enhancement in clerical productivity.

3.5.3 A Conclusion from the Case Review

To extensively review the EESI case study is beyond the scope of this thesis. However, interested readers are encouraged to refer to Plant et al. (1990).

The success of the case study has confirmed the expert system's ability to adapt to the complexities of the wage awards, and resolve the problems of manual timesheets. Plant et al. (1990, n.p) stated "knowledge base systems technology provided a means to rapidly prototype and implement the rules associated with a business process into the user application .... [and] the new technology augmented, it did not replace, the set of tools available to the developer".

The EESI methodology presented an overview of the development stages and is typical of project developments of today. As the knowledge engineering was undertaken by IBM, Plant et al. (1990) avoided discussing the issues and the techniques involved in knowledge acquisition.

Since the EESI pilot project, expert systems technology had been a major influence on new SECV projects as Plant et al. (1990) had discovered:

> The SECV believes that KBS [Knowledge Based Systems] technology can now move into the mainstream of programming techniques at the SECV [and] .... will actively investigate emerging AI [Artificial Intelligence] technology in order to realise equally substantial benefits. (n.p)

The statement suggested that the adoption of expert systems technologies to award implementation is worthy of further investigation.

CHAPTER 4

METHODOLOGY

Chapter Headings:

4.1 INTRODUCTION

The methodologies used in individual research areas are discussed in this chapter. The topics covered in this chapter include the development methodology for the entire project, the selection methodology for the expert system shell, and the methodology for knowledge acquisition.

4.2 EXPERT SYSTEM DEVELOPMENT METHODOLOGY

Typically, when embarking on a new software development project, one of the first steps to be taken into consideration is deciding the stages of the project development life-cycle[7]. A sad fact is that the technology and the development of expert systems is still in its infancy and as a result has yet to establish itself into industry standards. Guida et al. (1989, p.3) states that "the development of expert systems largely relies today on empirical methods and is not supported by sound and general methodologies. It is more like handicraft than engineering, and it lacks several of the desirable features of an industrial process (reliability, repeatability, work-sharing, cost estimation, quality assurance, etc)".

Thus, the framework of an expert system development methodology it seems, is depended upon the application domain and the specific organisation (Guida et al. 1989). The objectives of the PESWEA project have therefore dictated the relative size of the project and the stages to be incorporated into

[7] By the term *life-cycle* it is generally understood as a way of organising and distributing over the time the different activities needed to design, construct and maintain an artifact (Guida et al. 1989, p.8).

its development methodology. According to Guida et al. (1989, p.4), "the development process should be organised around a general, environment- and application-independent life cycle concept, whose structure and characteristics capture and take into account the peculiarities of expert system technology".

A set of *general engineering requirements* should be taken into account when designing an expert system development methodology, as listed by Guida et al. (1989, p.9):

- It should be *structured* and *modular*, i.e. it should support as far as possible (hierarchical) work decomposition into elementary components.
- It should be *complete*, i.e. it should support the designer in all aspects and phases of expert system development process, from early problem analysis to maintenance, and it should offer both technical support, and project management features.
- It should be *effective*, i.e. it should support easy planning and control of project development for what concerns activities, resources, results and time.
- It should be *efficient*, i.e. easy to apply without a sensible overhead for the project.
- It should be *practical*, i.e. easy to teach, transfer, and use in a large variety of different contexts.

The above requirements were specifically aimed at conceiving industrial expert system methodologies (Guida et al. 1989). Thus, a project like PESWEA is not required to satisfy all of the requirements.

Expert systems have frequently been viewed as a type of software product. As a result, conventional software development methodologies (or life-cycles) have been investigated and adopted by a number of authors and refined into the expert system life-cycle (Guida et al. 1989; Martin et al. 1988; Parsaye et al. 1988).

The resulting life-cycle of the PESWEA project was adapted from the models provided by the above three authors and was performed in these ordered sequence of steps:

i) Feasibility Analysis. The problem domain and the task to be performed by the expert system is identified, studied and analysed.

ii) Conceptual Design. Defining the conceptual structure of the system, and drafting specifications describing how the expert system will carry out the task.

iii) Shell Selection. Analysing available expert system shells on the market and the selection of an appropriate shell for the project.

iv) Knowledge Acquisition. The knowledge required by the expert system to perform a task is acquired from domain experts and reference sources.

v) Knowledge Representation. The knowledge gathered from the knowledge acquisition process is then formalised and represented within a "symbolic program" as a knowledge base to be executed by the inference engine.

vi) Validation. The opinions and views of domain experts and users, or a set of operational criteria are used to assess the degree of success the expert system has achieved.

vii) Technology Transfer and Maintenance. The expert system is transferred into an operational environment, and maintained over time to accommodate changing needs.

The likelihood of step (vii) of the life-cycle proceeding is yet uncertain. This decision was entrusted entirely to Westrail: whether they would like to proceed with the current shell, system organisation and technology or, in the light of the lessons learnt, abandon them in search of new technologies.

4.3 SHELL SELECTION CRITERIA

Selecting the right expert system development tool (or shell) is comparable to selecting conventional software products like database management systems. In fact, standard procedures already adopted by organisations for selecting software products can be used to select an appropriate expert system shell (Martin et al., 1988).

Prior to selecting an appropriate shell for the application, a number of technical points must first be addressed. This includes identifying the objectives of the system to be built and the functional requirements. Other selection criteria that should be taken into account includes vendor status and policies, and the shell capabilities (Martin et al., 1988; Brownstein et al., 1982).

The importance of thoroughly documenting the process to be used in the selection of shells was stressed by Martin et al. (1988). Figure 4.1 illustrates a possible shell selection process to be used in which readers are under no obligation to comply with every step (Martin et al. 1988, p.424). The first six selection steps are:

(1) Define the objectives. Involves identifying what the expert system should do (Martin et al. 1988, Brownstein et al. 1982).

(2) Determine the constraints. This process involves identifying the limits and restrictions of the project (Martin et al. 1988; Brownstein et al. 1982).

(3) List the assumptions. Involves documenting the user's and the developer's assumptions about the project (Martin et al., 1988).

(4) Obtain user requirements. This process involves obtaining and documenting what the user requires and expects of the expert system shell (Martin et al. 1988; Brownstein et al. 1982).

(5) Assess the environment. The environment in which the expert system is going to run must be assessed to determine how the expert system can adapt into existing equipment and operations (Martin et al. 1988).

(6) Research available tools/shells. The shells available on the market must then be investigated and compared to determine which is best suited to the problem domain and the requirements (Martin et al. 1988; Brownstein et al. 1982).

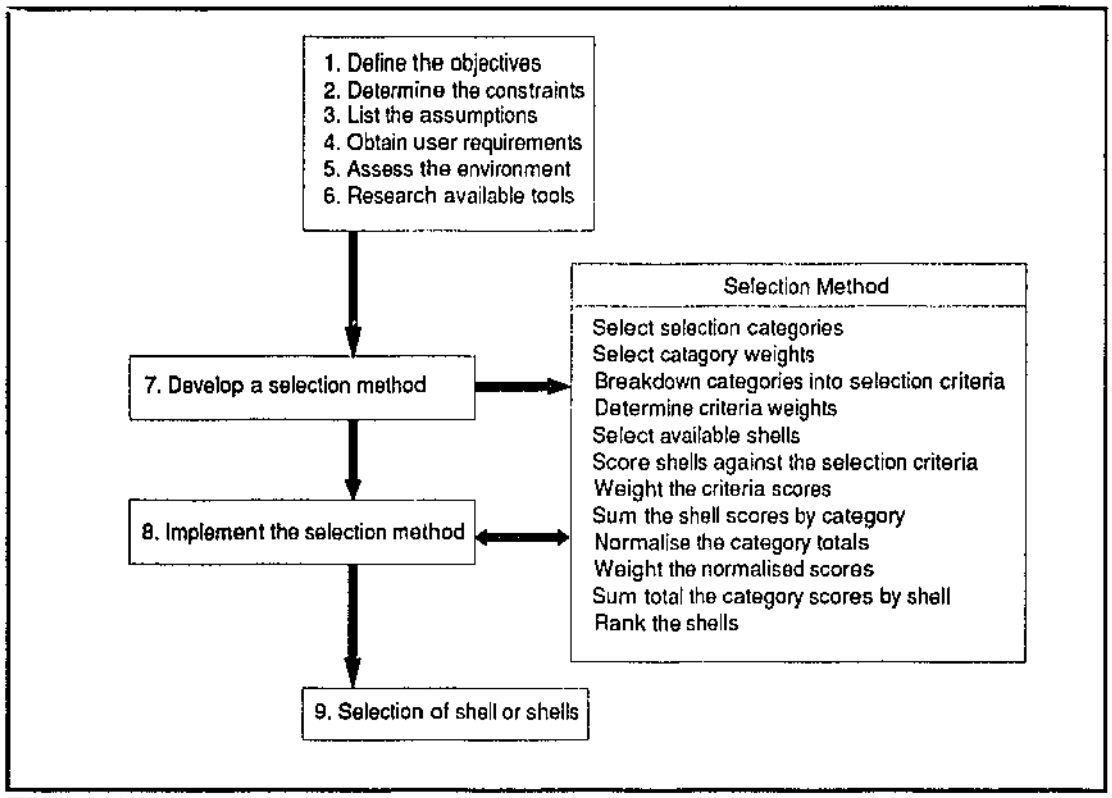

Figure 4.1 *The PESWEA shell selection process (adapted from Martin et al. 1988, p.424).*

The selection method as highlighted in Figure 4.1 gives rise to twelve steps (Martin et al. 1988, p.425):

1. Select selection categories.
2. Select category weights.
3. Break down categories into selection criteria.
4. Determine criteria weights.
5. Select available tools.
6. Score tools against criteria.
7. Weight the criteria scores.
8. Sum the tool scores by category.
9. Normalise the category totals.
10. Weight the normalised scores.
11. Sum the category scores by tool.
12. Rank the tools.

The first selection task involves identifying categories of the shells to be compared based on the functional requirements. An example of a category might be the knowledge representation of the shells. These categories are then weighted to determine the relative importance of each. If necessary, these categories can be further itemised and weighted into specific selection criteria. For example, knowledge representation can be itemised into schemes which includes rules, frames and the like. Available shells are then selected and scored against the listed criteria. The scores are weighted and totalled, and the shells are then ranked according to the scores (Martin et al. 1988).

The scoring of a shell is determined by its strength in supporting the project requirements. In order for the methodology to be effective, the scoring process would be performed with score sheets to assist in the shell comparisons. The final step is the selection of the appropriate shell based on the ranking as determined by the scores (Martin et al. 1988; Brownstein et al. 1982).

4.4 KNOWLEDGE ACQUISITION METHODOLOGY

As mentioned in Section 3.3.3, the process of knowledge acquisition is often characterised as being rapid prototyping. An effective knowledge acquisition methodology has yet to be established in the industry as many authors believed that the process of knowledge acquisition is inherently experimental possessing some degree of trial and error. However, the general process of knowledge acquisition can be identified by two factors, (Scott et al. 1991, p.13):

• ***Initial inquiry.*** In this preparatory stage of knowledge acquisition, a broad and concise overview is obtained to determine what the expert system is to do, how it will be used, and how it will be developed.

• ***Detailed investigation.*** Known as the "discovery stage", with an emphasis on focused details. The information gathered on how human experts perform their task should be understood by knowledge engineers so that they can emulate the same process into the expert system.

Interviews with human (or domain) experts play an important role in the process of knowledge acquisition. Roberts (1990, p.43) made a distinction between structured and unstructured interviews with citations made from Hoffman (1987):

> In order to add structure to an otherwise unstructured interview ... the knowledge engineer initially makes a first pass ... by analysing the available texts ... or by conducting an unstructured interview. The expert then goes over the first pass ... making comments ... recording

> this process is not necessary because the knowledge engineer can write changes and notes on a copy of the printout of the first pass ... the result is a second pass.

Roberts (1990, p.43) provided some interesting statistics gathered from Hoffman about the development time for some well-known expert systems. One of the first and most famous, MYCIN acquired "many years" in the making. Others like INTERNIST was developed in a period of ten years, while R1 "... took two man-years to develop by a team of about a dozen researchers and is still being refined". Of particular interest was the PUFF system which was developed in a period of less than ten weeks. Hoffman (cited in Roberts 1990, p.43) explained that "the likely reason for this brevity was that most of the rules were easily gleaned from archived data ... and only one week was spent interviewing the experts".

Keeping in mind the objective and the relatively small size of this project (PESWEA), a comprehensive knowledge acquisition process was considered to be quite unnecessary. Based on the findings of Roberts (1990), it was decided that the procedures involved in the knowledge acquisition process for this project would initially include a few semi-structured interview sessions with domain experts, after which further knowledge would be extracted from existing award manuals.

While acknowledging that some knowledge would be translated incorrectly and incompletely, the project does not aim to satisfy all the requirements of the awards. What was considered more important was the role of PESWEA in shedding light on expert system potentials for subsequent projects at Westrail.

Thus, the knowledge acquisition process for PESWEA emulates that of Roberts (1990, p.44), and would consist of:

• one or two semi-structured interviews (to discuss methods, timescales, basic terms and the like),

• extensive use of pre-existing texts, and once the initial prototype had been developed,

• many prototype feedback cycles, involving both structured dialogue and further references to the text.

Chapter 5

RESULTS OF STUDY

Chapter Headings:

5.1 INTRODUCTION

The results obtained in the research are discussed in relevant detail in this chapter. Each study is the result of the research questions detailed in Chapter One of the thesis.

5.2 STUDY ONE: SHELL SELECTION RESULTS

5.2.1 Objective

The objective of Study One was to search for an expert system shell that was available on the current software market and at best, met the constraints and the requirements of the PESWEA project. The appropriate shell would be used as a development tool in the project to assist in fulfilling the primary objectives of PESWEA.

5.2.2 Selection Requirements and Constraints

It was decided that the shell should run on personal computers for the following reasons:

• the cost of development and software on PC's were generally lower than on other machines.

• the problem domain was not sufficiently complex to warrant a more expensive machine.

• PC's were readily available as development and delivery machines.

• PC's made up a significant proportion of the hardware used at Westrail.

The shell selection criteria issued by Westrail were used to compare expert system shells currently available on the software market. To be considered for the final selection, the shells must have features that met the following *essential* selection criteria:

• Mathematical capacity for the calculation of duty hours;

• Personal Computer based with the ability to run under MS-DOS;

• Possession of excellent system interfacing facilities for communicating with external systems, programs or databases; and

• Unlimited rule capacity to accommodate large, complex rules.

In addition, the following *desirable* selection criteria were used for short-listing potential shells:

• Must have the potential to be used in further development projects;

• Good forward and backward chaining ability to allow maximum flexibility in rule modelling;

• Possession of good programming facilities to override the shell's automatic generation facilities. Must be Pascal- or C-based for familiarity, and to avoid the memory management problems of LISP-based products;

• Small memory requirements, preferably needing a maximum of 512K of main memory;

• Must be inexpensive to allow for trial runs. Any rejections due to the shell's inadequacy will therefore not be an expensive exercise;

• Possession of multiuser facilities to accommodate for multiaccess and enquiries;

• A separate run-time version must be available as a security feature, so that users cannot access or modify the knowledge-bases;

• Must be easy to learn requiring no more than seven training days; and

• Must have a good user interface (users will often have no prior experience with PC's, and very little experience with any computers).

5.2.3 Limitations

The process of Study One was delimited by:

• The majority of vendors specialising in expert system shells were located in the USA. Their products were either sold through a handful of software distributors in Australia or through direct order. This made the shell evaluation process more difficult when it came to collecting marketing and product details for the shells.

• Shells were not stocked and readily available through software distributors, preventing demonstrations of promising shells.

• New shells debuting on the market could not be reviewed due to the lack of product information and availability.

• Relative strengths of each shell to each criterion could not be accurately determined from the available literature.

5.2.4 Shell References

The shells evaluated were some of the current commercial expert system building tools in common use. The details of the nineteen shells evaluated were gathered from the works of Gevarter (1990b), Freedman (1990), Bielawski et al. (1988) and Bowerman et al. (1988). Thus the shells used in the evaluation were well established commercially. The drawback with this process was that many shell vendors have since merged with other companies which resulted in corporate name changes making contact difficult. However, many vendors continue to advertise their products with details of their location and contact numbers in computer journals. Good sources of vendor information are the *AI Expert* and the *IEEE Expert*.

5.2.5 Results of Study One

Table 5.1a and 5.1b represents the attributes of the nineteen shells being evaluated. The selection criteria were used as a basis for shell comparisons with the first five criteria weighted in order of importance in the tables. Due to the quality of the information from the available literature, not all aspects of the criteria could be used in the comparison. Each shell listed in the tables were marked according to how well they represented each criterion. A blank entry in the tables denote either a lack of information available for the shell, or the shell failed to meet the criterion.

NAME	ART	KEE	KNOW-LEDGE CRAFT	PICON	S.1	ESE VM/MVS	ENVI-SAGE	KES	M.1	NEX-PERT OBJECT
Math Calculation	X	X	X	X	X		X	X	X	X
MS-DOS on IBM								X	X	X
Specified Rule Limit									1000	2000
Forward & Backward chain	X	X	X	X	X	X	X	B	X	X
System interfacing	H	H	H	H	H	X	FC	F	X	FCD
Training							5	4		5
Documentation							190	300		500
Language of tool	LISP	LISP	LISP	LISP	C	Pascal	Pascal	C	C	C
Knowledge representation	RFrO	RFrO	RFrO	RFrO	R	R	R	R	R	RFr

X = Included H = Host computer F = Files C = Call facility D = Data base format conventions
R = Rules Fr = Frames O = Object-oriented E = Examples

Table 5.1a *The shell selection criteria*

The elimination process was initiated by comparing each shell to the first criterion: Math Calculation. It was surprising to learn that some shells did not possess such a basic feature which was vital to many applications. This narrowed the list of potential shells to sixteen, minus ESE, TIMM and Rulemaster.

The elimination continued with the next criterion: MS-DOS on IBM. This process saw ten shells succeeding with the essential requirements of *Math Calculations* and *IBM PC Support*, and the removal of these mainframe supporting shells: ART, KEE, Knowledge Craft, Picon, S.1 and Envisage.

NAME	PC+	EXSYS	EXPERT EDGE	ESP	INSIGHT 2+	TIMM	RULE-MASTER	KDS3	1st CLASS FUSION	
Math Calculation	X	X	X	X	X			X	X	
MS-DOS on IBM	X	X	X	X	X	X	X	X	X	
Specified Rule Limit	800	5000			2000			16000		
Forward & Backward chain	X	X	B	X	X	F	X	X	X	
System interfacing	FCD	FC	X	FCH	FCD		X	CD	FCD	
Training	10	3		3	5			5	1	
Documentation	1200	272		200	320			265	440	
Language of tool	LISP	C	C	PROLOG	Pascal	Fortran	C	Assem	Pascal	
Knowledge representation	RFr	R	R	RFr	R	RE	RE	FrE	RE	

Table 5.1b *The shell selection criteria continued.*

The third criterion: Specified Rule Limit, proved difficult to evaluate. A blank entry meant that there was no specified rule limit. However, as for the other shells, there was insufficient information to determine whether their specified rule limit was the maximum for one base module, and if so, could modules be chained to create more rules to form large systems. Hence an assumption was made at this stage that a specified rule limit only pertained to a single module. This saw the same ten shells proceed to the next criterion.

To provide development flexibility, the shells must have strong forward and backward chaining features. This criterion reduced the list to eight potential shells, and the elimination of these shells: KES and Expert Edge.

The remaining criteria could not be used as elimination factors but were instead used in Table 5.2 as scoring attributes. A scoring range of (0 - 5) was used to grade the remaining eight shells according to their relative strengths to each attribute: 0 for poor, and 5 for strongest or most favourable feature. The scores of each shell were added to form a total score with a maximum of 35. The total scores of each shell were normalised to form a percentage score. The final ranking of the shells were determined by their respective percentage scores.

Scoring in System Interfacing was determined by the shells' strengths in their language and database hooks, an information gathered from Gevarter (1990b, p.46). The scoring of the second attribute was determined by the end-user features possessed by each shell. The language C was regarded as the most important language for the fifth attribute, followed by Pascal, Lisp and Prolog. 1st-Class Fusion scored highly with the Experience attribute. This was owing to a basic demonstration model being available for evaluation. Shells with the ability to accommodate rules and examples in their knowledge-base scored highly for the sixth attribute. Rules and examples were regarded as favourable knowledge representation types for the users at Westrail.

Factor score (0 - 5) \ Name	KDS3	1st CLASS FUSION	IN-SIGHT 2+	NEXPERT OBJECT	PC+	EXSYS	ESP	M.1
System Interfacing	4	3	3	5	4	3	4	4
User Interfacing	4	3	4	5	4	4	3	3
Chaining strength	4	3	3	4	3	3	3	3
Training ease	3	5	3	3	1	4	4	0
Language for programming	4	4	4	5	3	5	3	5
Previous experience	0	3	0	0	0	0	0	0
Knowledge base type	4	5	3	3	3	3	3	3
SCORE (35)	23	26	20	25	18	22	20	18
NORM. RESULTS (%)	66	74	57	71	51	63	57	51
RANKING	3	1	5	2	6	4	5	6

Table 5.2 *The rank order of the most promising shells for PESWEA.*

The final ranking of potential shells were as follows:

1. **1st-Class Fusion.**
2. Nexpert Object.
3. KDS 3.
4. Exsys.

=5. Insight 2+

=5. ESP Frame Engine.

=6. PC+

=6. M.1.

5.2.6 Conclusion

The 1st-Class Fusion shell ranked first in the selection process and thus was chosen for the PESWEA project. However, 1st-Class Fusion only came first based on information gathered from the available literature, and the experience gained from a basic demonstration model. This leaves the full potential of the new shell unknown until "hands-on" experience is acquired. 1st-Class Fusion satisfied all criteria relevant to the PESWEA project only. The relative success of the PESWEA project would determine whether 1st-Class Fusion would be used in future Westrail projects. Nexpert Object and KDS 3 were also worthy of further investigation for future projects.

According to Table 5.2, 1st-Class Fusion scored average results for System interfacing, User interfacing and Chaining strength. Nexpert Object and KDS3, which came second and third in the ranking respectively, scored higher than 1st-Class Fusion in these factors. However, Fusion was easier to learn, and had more favourable knowledge representation types: rules and examples. The structure of the award application domain required an *Action* knowledge representation (see 2.2.2) for two reasons: although the award structure required some form of inheritance, a rule-based representation (see 3.3.7.1) was considered faster to learn; and cheaper shells possessed rule-based representation. Rules can simulate inheritance although not as elegantly as *Frames*. The *examples* representation, which belonged to the Action group, represented knowledge in a tabular format. As the awards possessed tabular data, examples was considered to be favourable by Westrail. Previous experience with a superseded 1st-Class shell helped Fusion to score in this category. Without this category, Fusion would have scored equally with KDS3. However, the limited time frame restricted the learning required for new shells. Thus, Fusion scored in this respect.

5.3 STUDY TWO: KNOWLEDGE REPRESENTATION

5.3.1 The 1st-Class shell

Based on the results from Study One, Edith Cowan University proceeded to order the 1st-Class Fusion shell from *AI Corp* of Massachusetts, USA. However, the shell received was an upgraded version re-named *1st-Class HT*, which had added features including hypertext. As HT was considerably more expensive than the superseded Fusion, the allocated university budget was insufficient to purchase the full package. The package received included a set of manuals and the HT programming environment at a cost of $910. Omitted from the package was the run-time program, a $2900 addition which allowed expert system builders to publish their knowledge-bases. The total cost of HT ($3810) far exceeded the budget of $2200 originally arranged for the Fusion package.

The 1st-Class HT product is an induction-based shell suited to situations where knowledge could be expressed in examples or derived from data in a tabular form. The HT is a general purpose shell with an interface resembling Lotus 1-2-3. 1st-Class HT was written in Microsoft Pascal and macro assembler. HT runs under DOS on an IBM Personal Computer with at least 256K of memory.

It offered sufficient flexibility in designing knowledge-bases through its unique forward- and backward-chaining techniques. Through chaining, several knowledge-bases could be linked together to form large expert system applications whose size was limited only by the amount of disk space available on the designated hardware. The chaining was implemented by

marking either factors (for backward chaining) or results (for forward chaining) within a knowledge base with a "#" symbol.

Knowledge rules were presented as decision trees. HT offered four different methods for creating a decision tree:

(1) ***Optimised method.*** From this option, HT automatically creates a decision tree from the examples entered. Eliminating unnecessary factors, a generalised rule is created that asks the least number of questions to reach a given result.

(2) ***Left-to-right method.*** Selecting this option generated a decision tree which asked questions based on the order the factors were entered into the matrix.

(3) ***Match method.*** This option avoids compiling the examples into a decision tree. With this method, the system simply works through each individual factor in an attempt to match examples to results provided by the user. This method is useful in running systems that are too complex to be compiled into a decision tree.

(4) ***Customisation method.*** If it is inappropriate to compile the knowledge into examples, this option allows the developer to make tailored decision trees. Creating rules using a decision tree is often tedious and at times inflexible compared to other simple rule-based systems. However, decision trees are seen by many critics as a graphical, user-friendly method of rule creation for non-programmers.

1st-Class HT can import data directly from spreadsheets, assign weights to particular examples, and provide statistics about the examples contained in a knowledge-base. Perhaps the most impressive facility included with HT is the ability to automatically generate portable codes. HT can take the graphical decision tree of any knowledge base and convert it into one of three forms:

- IF ... THEN production rule sets.
- Pascal source code.
- "C" source code.

Each translation feature is important in its own right, making HT a highly efficient and rapid prototyping tool for the PC. Perhaps the most interesting is the IF-THEN translation feature which allows an expert system builder to import rules translated from HT's decision trees into other expert system shells. Regrettably, HT's most impressive facility is also a disappointment in a sense that "foreign" source codes written in "C", Pascal or by other expert systems cannot be imported into HT and translated into decision trees.

There are six major screens in HT's programming environment:

Files: Shows the filenames of listed knowledge-bases, and allows for disk functions.

Definitions: Identified knowledge factors and results and defined.

Examples: An optional feature which allows the examples of a knowledge-base to be entered.

Methods: Allows the developer to select one of four methods for creating a rule.

Rule: A graphical representation of a decision tree generated from the examples is displayed in this screen. A customised tree can also be built to represent rules.

Advisor: A testing facility which allows the developer to simulate and test a rule.

An efficient feature of the HT is the developer's ability to add new factors or results while entering examples in the Examples screen or manipulating a rule in the Rule screen. With this facility the developer can avoid moving back and forth between screens to make substantial changes.

5.3.2 Knowledge Engineering with 1st-Class HT

5.3.2.1 Preparing the knowledge-base

Prior to building a knowledge-base, the basic properties of the knowledge-base must first be identified. After much experiment with HT, these steps have proved useful when constructing a knowledge-base:

Identify the structure of the knowledge-base. The first coherent step in building a knowledge-base is identifying what domains of the knowledge are to be included into the knowledge-base and what the goals of the system are. Furthermore, a decision has to be made on how the domains are to be represented as modules and chained to form a logically sound knowledge-base.

Identify the factors and their values. These are the attributes of a knowledge-base module used for classification or in executing the procedures of a task. Factors are usually the conditions, measurements, an observation or facts that determine the way things work. The values determine the state of a factor. For example, the factor OFF_DUTY may have the values [true,false]. If the rules were viewed in the IF...THEN form, these factors and their respective values would be used to construct the IF body of the clause.

Identify the results. In HT, the RESULT is a factor used in defining the conclusions of a knowledge. For a rule of the IF...THEN form, the result would succeed the THEN clause.

Should the task be Declarative (example based) or Procedural (rule based)? There are two distinctive ways in constructing a sub-task in HT. Declarative knowledge-bases declare known actions, values or situations, hence they can be constructed as examples in HT. Knowledge-bases built by example are automatically constructed into rules by HT. Conversely, the procedural approach allows for the manual construction of a rule in the rule-editor.

In determining when a task is to be built by examples or using rules, Thomas & Hapgood (1989) recommended these actions:

• Use examples if the system is a classification type, have historical or tabular data, or if it is a case scenario.

• Use rules if the system is to perform actions, loops, or if the knowledge-base is used to control the sequence or selection of other knowledge-bases, i.e. chaining.

5.3.2.2 Domain description

The award selected for the pilot project implementation was the *Railway Employees Award* (REA) of the *Marketing and Operating* section. The REA was considerably less complex than most other awards. Hence, the REA was determined by Westrail as being the most appropriate in meeting the primary objectives of the PESWEA study.

Regrettably, the late arrival of 1st-Class HT reduced the time period available in conducting comprehensive experiments in knowledge-base construction. A total of four sections were extracted from REA that were considered to be of a sufficient domain size. These sections were as follows:

(1) Guaranteed Week.

(2) Hours of Duty.

(3) Overtime Allowance.

(4) Saturday Time - Shift Workers.

The four sections were a representative collection of the Railway Employees Award (refer to Appendix A). They were chosen to be implemented as knowledge-bases as each section had conditions that allow for the interaction of one another. This was considered as a challenging factor for the chaining mechanisms of 1st-Class HT. Furthermore, the four sections of the REA held sufficient complexity for rule construction.

The techniques for fabricating knowledge-bases in HT will be illustrated in the following segments. A knowledge-base module known as "MonHrs" held the rules pertaining to the *Monday hours* which were related to the *Overtime Allowance* section of the REA. *MonHrs* was designed using

many of the facilities available in HT. Hence it was considered appropriate for discussion.

Figure 5.1 illustrates the relationship between each knowledge-base module as implemented in HT. All KB modules are forward chained to the module *Hrsdut* (Hours of duty). In this configuration, the results of the sub-modules are passed into *Hrsdut* for further processing. The factors and the respective values of *Hrsdut* are defined as follows:

- **#Monhrs** Factor to backward-chain to module Monhrs to calculate the hours of duty for Monday and the respective overtime allowances.
- **#Tuehrs** Backward-chain to module Tuehrs.
- **#Wedhrs** Backward-chain to module Wedhrs.
- **#Thurhrs** Backward-chain to module Thurhrs.
- **#Frihrs** Backward-chain to module Frihrs.
- **#Sathrs** Backward-chain to module Sathrs for the calculation of the Saturday penalties.
- **Tothrswk** This factor calculates the total hours for the week from data received from the external modules.
- **#OT2Use** This factor will backward-chain to the module OT2Use for the daily and weekly overtime comparison. The highest overtime is used.
- **Results** The results factor of *Hrsdut* holds the values for the Guaranteed Hours (Grnthrs), the Six Hour Creditation (6credit) and the Two Hour Creditation (2credit). Only one of these values will be selected depending on the results of the total hours calculated.

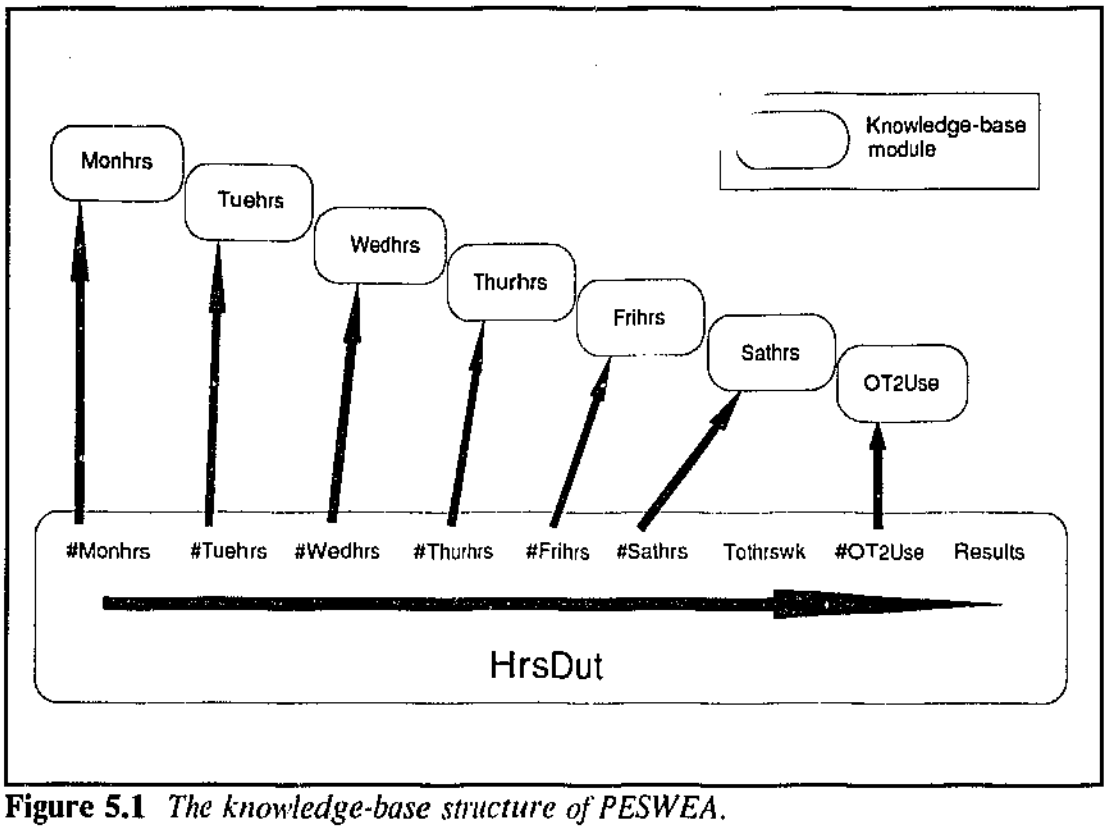

Figure 5.1 *The knowledge-base structure of PESWEA.*

A full discussion of the rule structure for *Hrsdut* is beyond the scope of this exercise, however the following section will examine the techniques used to implement a sample knowledge (MonHrs) into HT.

5.3.2.3 Knowledge-base construction in HT

Upon loading HT, the first screen to be encountered was the *Files* screen. This purpose-built screen lists the knowledge-base files available that were previously built with HT, as illustrated in Figure 5.2. Continuing with an existing knowledge-base was invoked by selecting the command "Get", then selecting the required file with the cursor keys. New knowledge-bases are created by selecting "New" from the menu. To develop the *Monday Hours* knowledge-base, a file called "MonHrs" was created throught the "New" command. The function of the keys "F9" and "F10" moves backwards and forwards respectively between the menu screens.

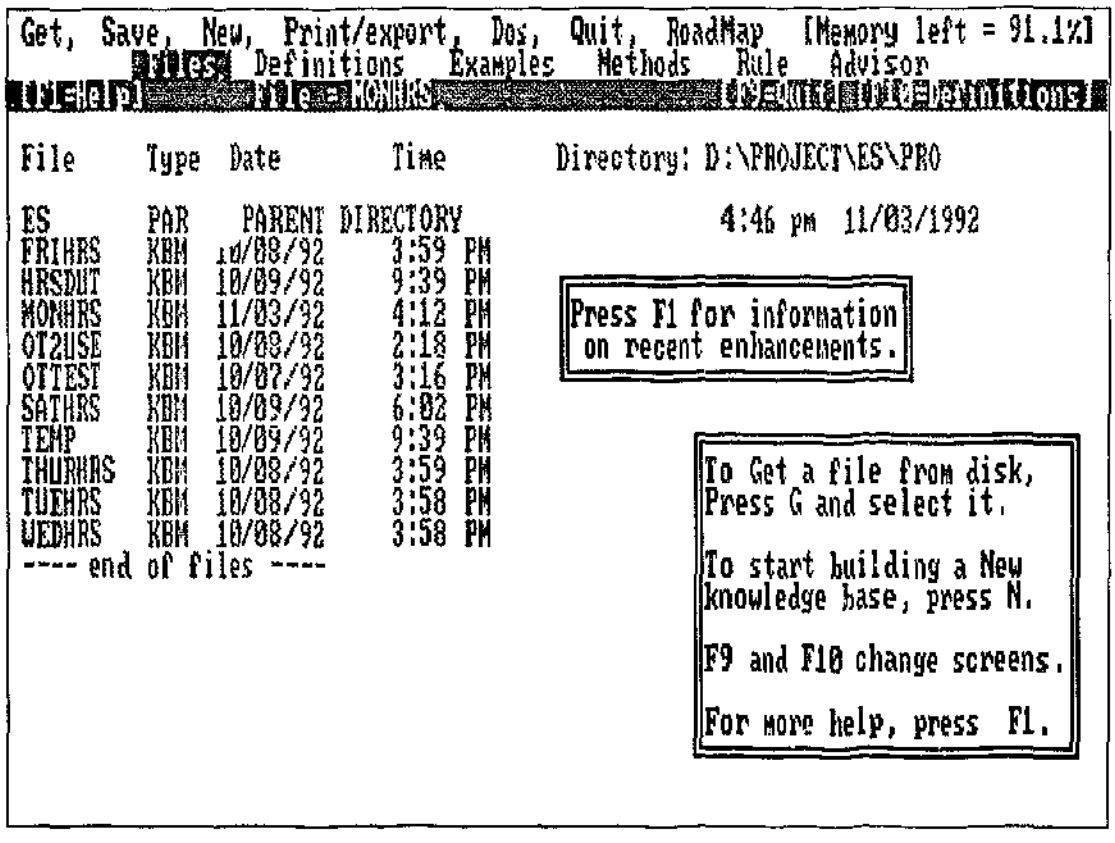

Figure 5.2 *The Files screen of 1st-Class HT*

Knowledge Entry

The factors identified for the *MonHrs* module were:

HrsWrk (Hours worked): The total number of hours worked in a shift was to be calculated and entered manually by the user.

Results (Results): The daily overtime allowance on shifts worked in excess of eight hours, Monday to Friday, was paid in accordance with the scale values held in this factor. According to the total hours worked, the rule would determine which one of the following result values would be used when returning to the parent knowledge-base: *Back2Hrsdut* (return to parent KB with no change); *Timehalf 1-12* (return to parent KB with a 50% value); or *Double 1-12* (return to parent KB with a 100% value).

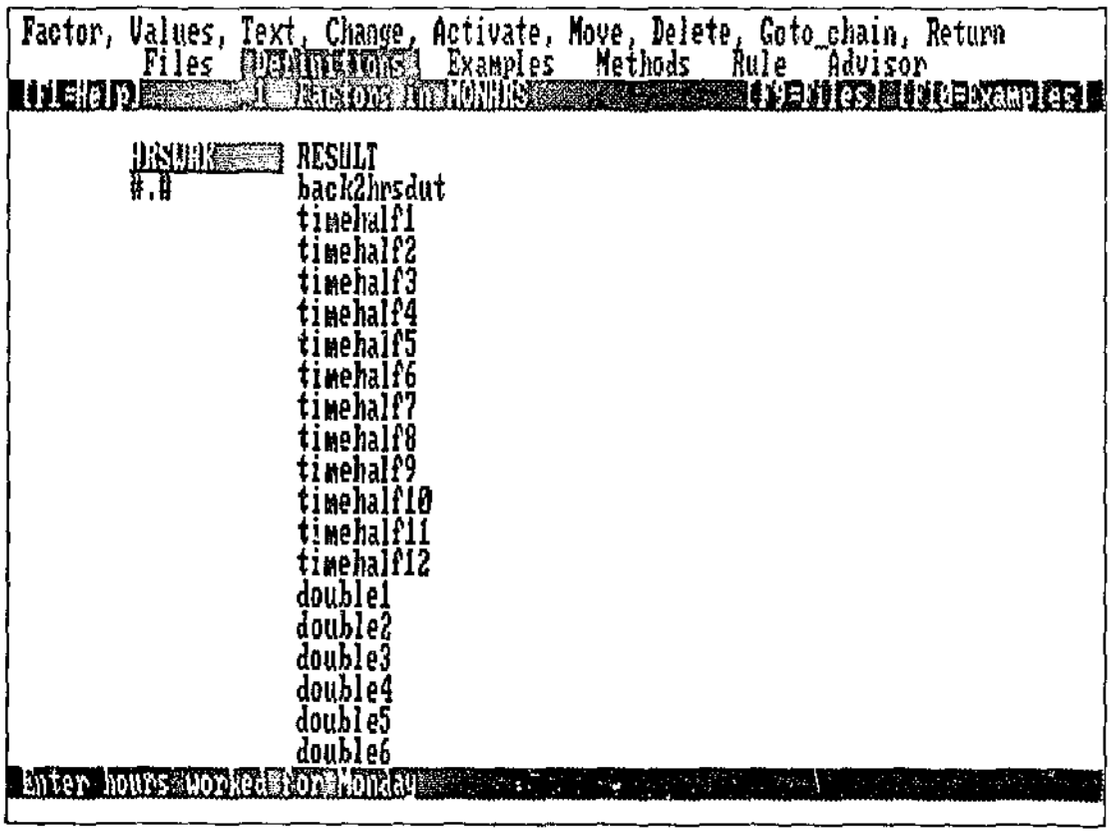

Figure 5.3 *Factors and values are defined in the Definition screen of HT.*

Figure 5.3 illustrates the Definition screen where the factors and the values for *MonHrs* were defined. Entering a factor or a value was invoked by the selection of the appropriate menu command. The "Text" command allows further editing of each factor or value. By invoking this command, developers have the ability to include mathematics to be used by the expert system in calculations or, create hypertext or plain text for system users.

Creating examples

The *MonHrs* domain was considered appropriate to be entered as examples as it possessed substantial tabular information. Figure 5.4 illustrates the Examples screen with the tabular details of *MonHrs* entered.

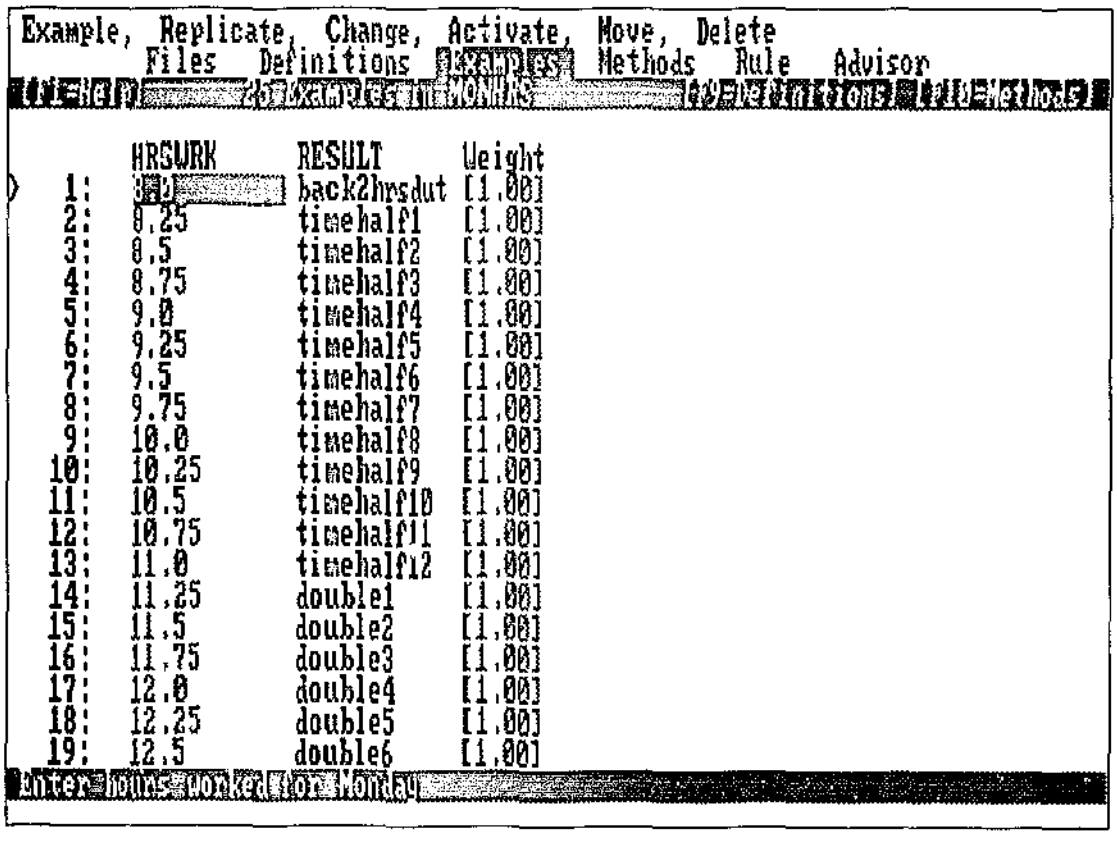

Figure 5.4 *The user has an option of entering examples to be generated into a decision tree by HT's induction facility. The table can be viewed as a decision table or a spreadsheet much like Lotus 1-2-3.*

A certainty weight of 1.00 had been assigned to each example. HT handled certainty factors by assigning weights to particular examples in the knowledge-base. This weighting feature allowed the developer to have several identical examples in the knowledge-base, each with a different weight. Thus, when a rule was induced from this type of knowledge-base, the program would automatically rank the results according to the weighting scheme.

Rule induction

The selection of the Optimized method provoked HT to induce an "efficient" rule based on the examples provided in the Examples screen. Figures 5.5 and 5.6 illustrates how HT induced the rules from the examples entered at the Examples screen, resulting in a hierarchy of binary thresholds. If the rule was designed using the Customised method (that is, bypassing the Examples screen), a smaller decision tree may have been created. The current decision tree was the result of the induction algorithm used by HT. Although cumbersome in nature, the induced decision tree was focused and accurate.

Tracing

Tracing in 1st-Class HT was usually accomplished by simply looking at the rule to see the decision path. If a problem path was noted in the rule, then the developer could *mark* the rule in the trouble spot. HT would then proceed to mark the corresponding example in the Examples screen. The developer could then edit the example base accordingly and generate a new rule which would correct the problem.

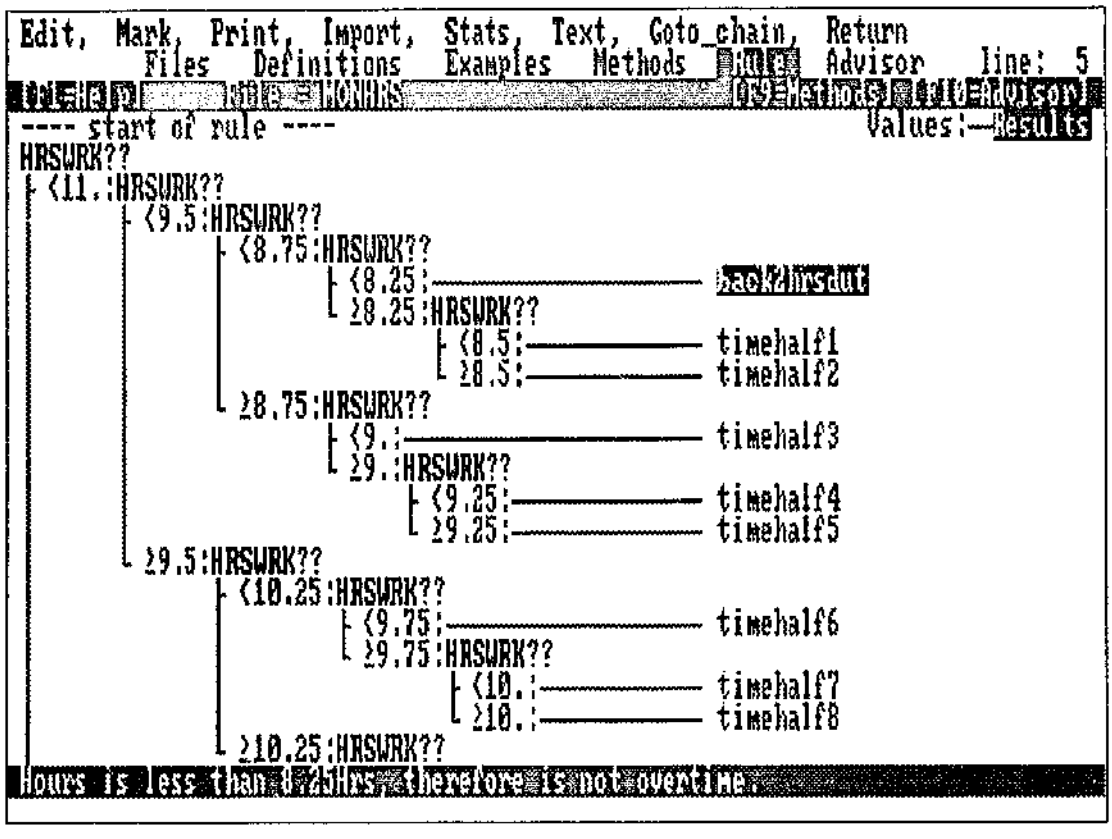

Figure 5.5 *Rules induced from the examples of MonHrs. This illustrates the top half of the decision tree.*

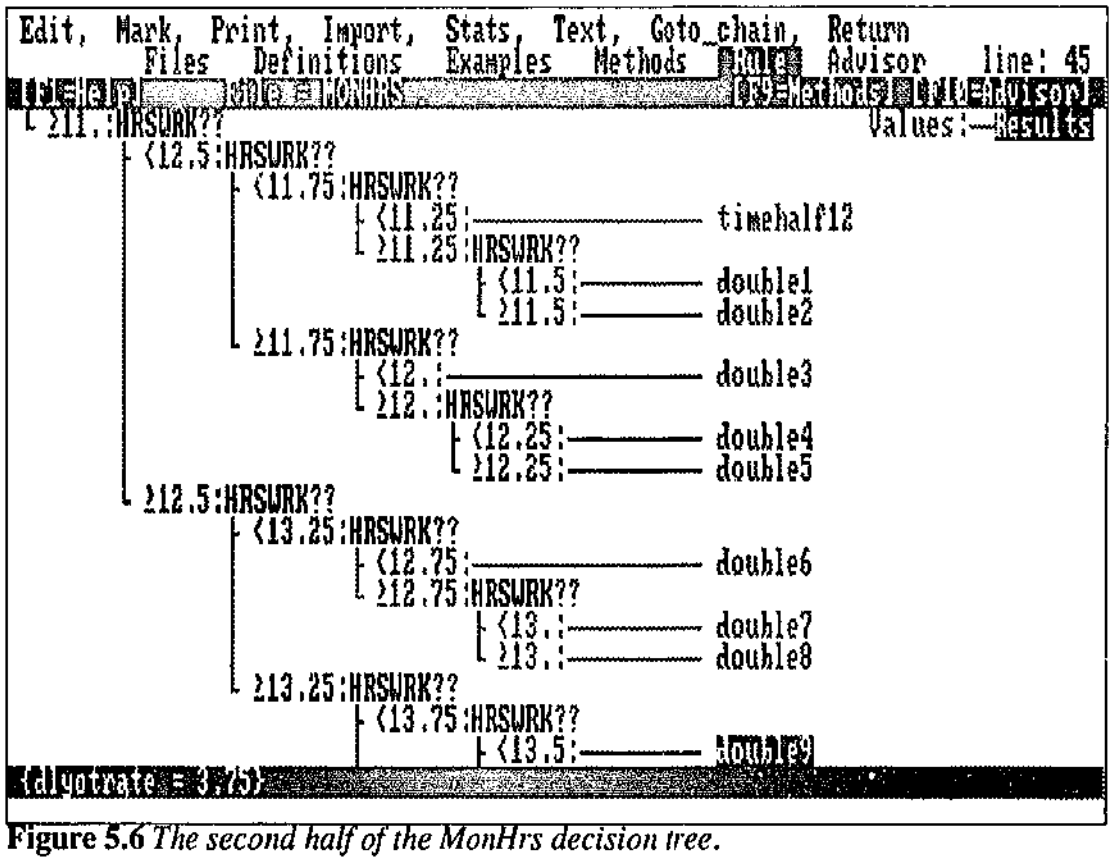

Figure 5.6 *The second half of the MonHrs decision tree.*

5.3.2.4 Using Decision Tables

Decision tables provide an effective way to collect the required knowledge to be represented as examples. Figure 5.7 provides a summary of the decision-making process of *MonHrs*. When examining the knowledge contained in the table in Figure 5.7, each column could be thought of as an example. The attributes *Hrswrk* (Hours worked) and *Result* could be considered as factors in HT's Definition or Examples screen. The knowledge involved in this form of decision-making process could also be represented in a rule format. For example, the third row of the table could be read as "IF the Hours Worked in one shift is 8 and a half hours, THEN the result of the Daily Overtime allowance is a quarter". Note that "8.25" was added into the table as an individual value. As the compilation of the table was based on an award rule, having "8.25" as an individual value would later assist in the knowledge validation process.

IF				THEN	
FACTOR	Hrswrk	?	?	Result	Value name
VALUES OF ATTRIBUTE	#.#			#.#	
EXAMPLES OF RULES	<8.25			0	back2hrsdut
	8.25			0	timehalf 1
	8.5			0.25	timehalf 2
	8.75			0.5	timehalf 3
	.			.	.
	.			.	.
	.			.	.
	10.75			1.5	timehalf 11
	11			1.5	timehalf 12
	11.25			1.75	double 1
	11.5			2	double 2
	.			.	.
	.			.	.
	.			.	.
	14			4.5	double 12

Figure 5.7 *A decision table aid used in constructing MonHrs.*

Note, however that the decision table was not designed to be restricted to the listed examples. Further combinations could be made by adding more factors and values to the table. For example, if a hypothetical factor *HAH* (Held Away from Home allowance) and its related values were added to the right of the factor *Hrswrk*, a new rule could be formed: "IF the Hours Worked in one shift is 8 and a half hours, AND the Held Away from Home allowance is ... THEN the result of the Daily Overtime allowance is ...".

Furthermore, by adding more values to each factor in the decision table, a large set of rules could be formed, each covering a different set of combinations. Once a knowledge is created using a decision table and entered as examples into HT, the inductive system would develop a decision tree that would work through the attributes efficiently, asking questions and finally making a recommendation.

Figure 5.8 illustrates a representative set of production rules translated from the decision tree of the knowledge-base *MonHrs*. Included with HT was a facility which allowed developers to translate their decision trees into production rules, Pascal code, or "C" code.

The production rules of Figure 5.8 were the result of the decision tree illustrated in Figure 5.5 and Figure 5.6. The induction algorithm of HT identified thresholds that would reduce each rule to their respective results. When the resulting decision tree was translated to production rules, the outcome was the recurrence of a single factor in each rule. Note that although clumsy in nature, each rule was technically correct.

Interestingly, if the decision tree was constructed using HT's Customise method, the resulting tree and the production rules would be more

refined. For example, Rule Result 1 of Figure 5.8 could easily be trimmed to:

Rule Result 1
If Hrswrk < 8.25
Then Result is back2hrsdut

Interested readers are encouraged to refer to Harmon et al. (1988, p.94) who provided an excellent case of decision tables and their relationship with inductive shells like 1st-Class HT.

```
RULE   RESULT 1
IF     HRSWRK          < 11.00
AND    HRSWRK          < 9.50
AND    HRSWRK          < 8.75
AND    HRSWRK          < 8.25
THEN   RESULT IS back2hrsdut

RULE   RESULT 2
IF     HRSWRK          < 11.00
AND    HRSWRK          < 9.50
AND    HRSWRK          < 8.75
AND    HRSWRK          >= 8.25
AND    HRSWRK          < 8.50
THEN   RESULT IS timehalf1

RULE   RESULT 3
IF     HRSWRK          < 11.00
AND    HRSWRK          < 9.50
AND    HRSWRK          < 8.75
AND    HRSWRK          >= 8.25
AND    HRSWRK          >= 8.50
THEN   RESULT IS timehalf2

       .
       .
       .

RULE   RESULT 23
IF     HRSWRK          >= 11.00
AND    HRSWRK          >= 12.50
AND    HRSWRK          >= 13.25
AND    HRSWRK          < 13.75
AND    HRSWRK          >= 13.50
THEN   RESULT IS double10

RULE   RESULT 24
IF     HRSWRK          >= 11.00
AND    HRSWRK          >= 12.50
AND    HRSWRK          >= 13.25
AND    HRSWRK          >= 13.75
AND    HRSWRK          < 14.00
THEN   RESULT IS double11

RULE   RESULT 25
IF     HRSWRK          >= 11.00
AND    HRSWRK          >= 12.50
AND    HRSWRK          >= 13.25
AND    HRSWRK          >= 13.75
AND    HRSWRK          >= 14.00
THEN   RESULT IS double12
```

Figure 5.8 *The production rules translated from the MonHrs decision tree.*

5.3.3 Verification of the Acquired Knowledge

Two sessions were staged where the information acquired for PESWEA was exposed to the experts of Westrail, both participants and non-participants, for thorough verification. The aims of the first demonstration were two-fold:

(1) To allow the domain expert to familiarise himself with the way the knowledge was represented explicitly.
(2) To verify the validity of the knowledge-bases.

The first session was conducted on a one-to-one basis. The demonstration included a test-run of the knowledge-bases under HT's Advisor facility, and a step-by-step assessment of each knowledge rule structure under HT's programming environment. The feedback from this session, including criticisms and contributions, was carefully recorded and used to enhance the system for a second demonstration to senior consultants and experts. The aim of the second demonstration was:

- To demonstrate the applicability of expert system technology to the awards.

The agenda of the second session was an iteration of the first. However, the discussions between those present at the meeting were focused on the credibility of the expert system technology.

5.3.4 Conclusion

The feedbacks from the second session were positive, and in favour of the expert system technology. The general view was that expert systems had the potential for award implementation, and was worthy of further investigations in future projects. This move may result in a full integration of expert system technology with Westrail's existing applications.

However, some individuals in the group felt slightly unconvinced, and it was with regret that the small time frame available did not permit further investigations into the PESWEA project. Whatever the outcome, the PESWEA project had laid the crucial foundation in expert systems development where future projects would be built in-house at Westrail.

The future of 1st-Class HT in future Westrail projects was indecisive. The programming structure of HT was deemed favourable by several domain experts, but its relative flexibility and power were dubious. HT was a sufficient tool in this project development due to the small size of PESWEA. Whether HT could be utilised in larger projects required further investigations.

Based on an informal survey of reports of expert system developments in Australia, 1st-Class HT was not a highly regarded and utilised tool. 1st-Class was most popular amongst smaller, personal computer-based developments in which the developers were non-programmers. Thus, it must be stressed that the original intentions of the manufacturer, and the design specifications of 1st-Class were focused as such. From this point of view, the potentially large future projects of Westrail would require a more capable shell. The type of shell suitable (PC- or mainframe-based) would depend on how much equipment re-tooling Westrail was prepared to undergo.

PESWEA was an *Interpretation* expert system (see 2.1.5 for a clarification). PESWEA analysed observables (that is, the working hours entered into the system) to determine its meaning and to conclude with a solution (the total hours and penalties), hence its Interpretation classification. The 1st-Class HT shell incorporated *Rules* and *Examples* to represent its knowledge-bases. As a result, PESWEA belonged to the *Actions* class of knowledge representation (see 2.2.2). Actions are one of the most common forms of knowledge representation available on PC-based shells. The *Forward* and *Backward* chaining inference engine mechanisms (see 2.2.3) were extensively used to link each knowledge-base module in PESWEA. Furthermore, another inference engine mechanism, *Math Calculations*, provided the mathematics required for the data interpretation.

5.4 STUDY THREE: SYSTEM INTERFACING ROUTINES

5.4.1 Interfacing With Databases

One of the requirements of PESWEA was to interface to an external system to receive data for processing. *DataFlex* was a database management system in use for the current timekeeping system at Westrail. A requirement for the PESWEA study was to analyse the potential of expert systems integration to a front-end DataFlex user system. The general concept was of an inferential expert system engine to the data keyed into DataFlex.

1st-Class HT had extensive integration capability with other programs and databases. HT could read and write data files that were created with Lotus 1-2-3 or dBASE and other programs once they have been written into an ASCII report form. HT could also import external data to be used as examples within a knowledge-base. Furthermore, HT took advantage of DOS ERROR_LEVEL numbers.

HT was specially designed to communicate directly with Lotus 1-2-3 or dBASE. The dBASE commands recognised by HT included:

- SEEK (to find a desired record).
- GET (to read a field value from a record).
- PUT (to update a field value in the record).
- APPEND (to add a new record to the database).

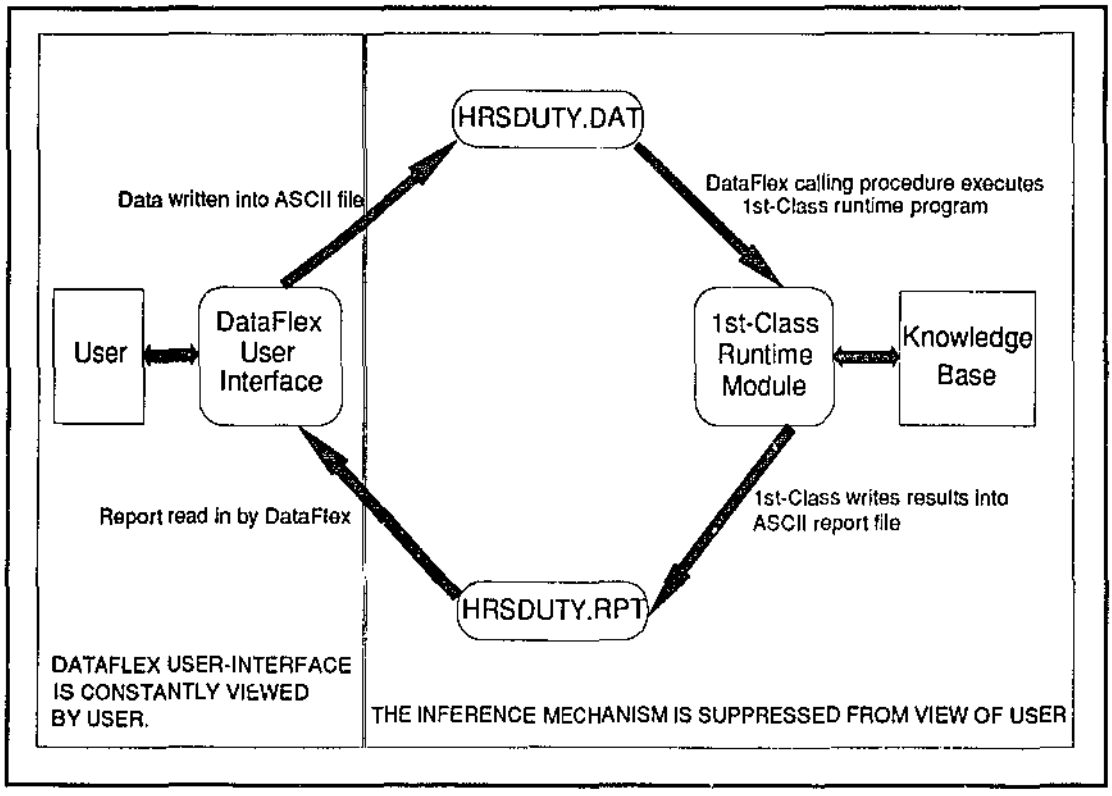

Figure 5.9 *The interfacing architecture of PESWEA.*

The above commands were not sufficient for the PESWEA problem domain. Further interfacing which required the transition of mass data could only be achieved through ASCII files. The interfacing architecture of PESWEA is illustrated in Figure 5.9. Data to be read by HT was written into an ASCII file with a ".DAT" extension. The results generated by HT were imported into DataFlex via an ASCII report file with a ".RPT" extension. The necessary data to be read by HT were imported with the command:

{READ HRSDUTY.DAT ALL}

The above command would invoke HT to *read all* the data contained in the file *HRSDUTY.DAT*. Consequently, exporting data to DataFlex was invoked by the command:

{CALL REPORT HRSDUTY.RPT (TOTHRSWK), (TOTHRS)}

This command would prompt HT to *call* the *report* utility to create the file *HRSDUTY.RPT* containing the results of the variables *TOTHRSWK* and *TOTHRS*. For HT to understand the information delivered by an external program, a dedicated routine had to be formulated by a programmer. HT would only comprehend information in the form:

VARIABLE = VALUE

Thus, the routine to export the required values of the variables in DataFlex were as follows:

```
OUTFILE "HRSDUTY.DAT"
        WRITELN "MONHRS = " FILENAME.MONHRS
        WRITELN "VARIABLE = " FILENAME.VARIABLE
OUTCLOSE
```

The above routine reads: direct output to the file *HRSDUTY.DAT* with the literal string MONHRS = followed by the value of the variable FILENAME.MONHRS, and repeat as required with other variables. The routine to import values into DataFlex turned out to be more elegant:

```
DIRECT_INPUT "HRSDUTY.RPT"
    [SEQEOF] ERROR 75 "HRSDUTY.RPT"
    [SEQEOF] ABORT
READLN FILENAME.TOTHRSWK FILENAME.TOTHRS
```

This routine informs DataFlex to prepare the file HRSDUTY.RPT for sequential input in which the values of the variables FILENAME.TOTHRSWK and FILENAME.TOTHRS would be read. Finally, the inference engine paradigm required DataFlex to execute the HT knowledge-bases during run-time. This was made possible with a dedicated DataFlex routine:

```
RUNPROGRAM WAIT "FRUN" "HRSDUT/X"
```

This routine would pause the DataFlex program currently running, execute the HT run-time program FRUN.EXE and pass it the argument HRSDUT/X; upon termination of FRUN, return to the DataFlex program and resume execution. The WAIT option would cause the DataFlex program to temporarily terminate, and stay resident while executing FRUN. The parameter HRSDUT/X informs FRUN to access the knowledge-base HRSDUT while suppressing the Advisor facility screen from view.

5.4.2 Multiuser Access

A requirement in the PESWEA study was to analyse the multiple user potentials of the shell. Regrettably, 1st-Class HT lacked the facilities to allow multiple user access to its knowledge-bases. A proposal put forward to the Westrail domain experts and consultants was of a model that isolated HT and

its knowledge-bases on a single computer. Established copies of DataFlex on external computers could then access the knowledge-bases via its own multiuser facilities.

The drawback with this scheme was that the DataFlex facility allowed one user access at any one given time. Simultaneous access by two or more users would either result in a queue or a suppression of others. Thus, the model was deemed unacceptable. If 1st-Class HT had to be utilized in future projects, multiple copies of the run-time program would have to be purchased and installed on individual computer units in the interest of multiple users.

5.4.3 Conclusion

The formulation of ASCII files allowed larger amounts of information to be passed between systems, but at the expense of processing speed. Virtual files, however, could be created in the computer memory which would by-pass the disk access resulting in increased performance.

Whether 1st-Class HT was inferior in systems interfacing could not be determined as other shells were not available for a comparison. HT possessed powerful commands to interface with an external database. However, HT's power in system-to-system interfacing remained dubious. This could be the consequence of a rising trend in the expert systems industry to integrate expert systems with a database, resulting in an *Intelligent Database*. At present, the power of system-to-system interfacing belonged to many expensive shells with dedicated interfacing features. Thus, reputable interfacing shells like Nexpert Object are worth further investigation.

Multiaccess capabilities were a feature omitted from many PC-based shells including 1st-Class HT. This feature was not highly publicised nor utilised in PC-based expert system applications, hence the potentials of other PC-based shells were not known. Conversely, an abundance of mainframe-based shells have dedicated multiuser facilities due to their natural environment. As mentioned in Section 5.4.2, multiaccess for HT could be accomplished either through the DataFlex environment or through the purchase of multiple copies of the HT runtime programs. This was a decision left to Westrail.

HT incorporated a user interface designed for developers with little programming experience. Thus, this made the HT programming environment favourable to normal users who could manipulate the knowledge-bases without the assistance of a programmer, providing they have access to the HT programming environment. However, HT did not provide sufficient flexibility for developers to create a customised runtime user interface. This required the assistance of an external system (e.g. DataFlex). PESWEA used the *Inter-system Communication* approach to systems interfacing (see 3.4.2.3). Model (b) of Figure 3.8 best represented the systems interfacing structure utilised by PESWEA in which the DBMS (in this case, DataFlex), dominated the concentration of processing and control.

CHAPTER 6

SUMMARY

Chapter Headings:

6.1 GENERALISATION OF RESULTS

6.1.1 Shell selection

The evaluation of expert system shells were made possible through literature. Demonstrations of shells were made impractical due to a lack of vendor support.

Table 5.1a and 5.1b lists the shells used in the evaluation process. The attributes of the shells are compared to a list of criteria in the tables. Eliminations of shells are initiated by the shells' ability to meet each criterion. Shells that pass the criteria short-listing are then ranked in accordance to their strengths in meeting a second set of criteria (see Table 5.2). 1st-Class Fusion ranked first, followed by Nexpert Object and KDS 3. 1st-Class succeeded in meeting the criteria for the PESWEA project.

1st-Class Fusion did not score highly in all respects, but did possess favourable factors including ease of learning and knowledge representation types. Rules are extensively used in cheaper shells. In addition, Rules and Examples are considered appropriate to represent the award domain. Rules can represent knowledge that perform actions, while examples are best for representing tabular knowledge data.

6.1.2 Knowledge representation

The acquired 1st-Class HT product is an induction-based shell suited to situations where knowledge can be expressed in examples or derived from data in a tabular form. It offered sufficient flexibility in designing knowledge-bases through its forward- and backward-chaining techniques. Through chaining, several knowledge-bases can be linked together to form large expert system applications whose size is limited only by the amount of disk space available on the designated hardware.

The knowledge rules of 1st-Class HT were represented as decision trees. There are four methods of creating a decision tree:

- ***Optimised method*** (an efficient automatic generation).

- ***Left-to-right method*** (ordered automatic generation).

- ***Match method*** (examples are matched to results. The decision tree is not created by this method).

- ***Customisation method*** (to generate tailor-made tree structures).

Rules built in 1st-Class HT can be translated into one of three source codes for portability:

- IF ... THEN production rule sets.
- Pascal source code.
- "C" source code.

There are four initial steps prior to building a knowledge-base (see 5.3.2). These steps require thorough knowledge analysis and design before proceeding to use the 1st-Class HT programming environment. A decision has to be made as to whether the knowledge are to be presented as examples or rules in HT. Tabular data are best represented as examples, while actions are best represented as rules.

The four sections of the Railway Employees Award implemented into HT (see 5.3.2.2) are the:

- Guaranteed Week.
- Hours of Duty.
- Overtime Allowance.
- Saturday Time - Shift Workers.

Software engineering CASE tools known as *Decision Tables* are a valuable aid in the construction of examples (see 5.3.2.4). This claim is supported by Francioni et al. (1988) and Harmon et al. (1988). At times the rules generated from examples by HT are not elegant but are accurate and practical. This is due to the binary threshold nature of the decision trees and HT's induction algorithm.

The authenticity of the knowledge-bases built in HT were verified by experts and consultants at Westrail in two individual sessions (see 5.3.3). The knowledge-bases presented are not highly complex and do not cover all technical aspects of the award. However the nature of the knowledge-bases are sufficient to fulfil the objectives of the PESWEA project. The general view from Westrail are positive towards the expert system technology. The chances of expert system integration into the current Westrail organisation are extremely high. However the future of 1st-Class HT in future projects remains in the hands of Westrail.

PESWEA was an *Interpretation* system incorporating the *Action* form of knowledge representation. The inference mechanisms utilised include *Forward* and *Backward* chaining to link modules, and *Math Calculations* to generate results.

6.1.3 System interfacing

The most practical method of transfering mass data across systems in 1st-Class HT is via ASCII files. This method was accomplished at a loss of processing speed. However, virtual memory files can be created to by-pass the disk access required.

Whether this was the best form of system interfacing offered by expert systems could not be determined as there were no other shells available for a comparison. HT allows special routines to be written in "C" or Pascal to enhance a variety of the shell's performances. However, the small time frame available did not permit a comprehensive design, testing and comparison of different interfacing techniques. The ASCII file routine was chosen as it was a complete facility available on HT, and the process could be designed by potential Westrail users with little programming experience.

PESWEA utilised the *Inter-system Communication* approach to systems interfacing.The concentration of processing and control was dominated by the DataFlex DBMS. This was one of three forms of control as depicted in Figure 3.8.

6.1.4 Generalised conclusions

The following conclusions can be made because of this study:

• 1st-Class HT is the most appropriate shell to be used in the PESWEA study based on the evaluation (see 5.2.5). Furthermore, it is the most appropriate shell to expose expert system technology to newcomers who want to proceed in this field.

• Expert system technology has the potentials to accommodate the complexities of the award structure. This claim has been proven through this study, and the study conducted by SECV in Victoria, Australia (see 3.5).

• 1st-Class HT has provided an insight into what can be achieved in the mass transition of data between expert systems and conventional systems.

• The technical and business objectives of PESWEA have been achieved with success.

6.2 LIMITATIONS REVISITED

The results of this study are limited due to:

• The size of the pilot project was reduced to ensure satisfactory completion within the limited time frame.

• While acknowledging the assistance offered and provided by Westrail, this study was essentially a one-person, supervised learning, project implementation.

• The project was sized to satisfy the objectives of PESWEA only.

6.3 LESSONS LEARNT FROM THIS STUDY

There are many lessons learnt from this study that do not relate to the research questions of the PESWEA project. They are:

• Apart from gaining an experience in expert systems technology, this study has shown what is required in the management of software and

systems. Constant business and public relations have enhanced personal confidence.

• In hindsight, better contacts with expert system user-groups and vendors should have been developed to exploit the latest in expert systems technology.

• Exploitations of new technology require a dedicated effort from all involved for the project to succeed.

6.4 FUTURE RESEARCH DIRECTIONS

Due to the success of the PESWEA project in meeting the objectives, an analysis for the development of a production system is currently underway at Westrail. To be included with the analysis is an effort to evaluate suitable expert system shells that are appropriate to the requirements of potentially large systems, and the organisational structure of Westrail. It is expected that the continued use of expert systems technology will be considered at Westrail as further business requirements are identified.

6.5 CONCLUSION

In this study, expert systems have proved to be a promising technology in the implementation of the Westrail awards. The study has further confirmed SECV's own findings into the application of expert system techniques in award implementation. The success of the PESWEA project has been due to the cooperative effort between the information technology staff and the domain experts of Westrail, and Edith Cowan University. Although the size of the PESWEA project is on a small scale, the knowledge gained have been invaluable to all involved.

The key technology focused was expert systems technology, in this case, 1st-Class HT by AI Corp. HT was chosen for the PESWEA project as it succeeded in meeting the selection criteria. However, this does not mean that HT was better than the other shells that made up the final ranking. The general perception was that Nexpert Object and KDS3 possessed powerful features that were more favourable to HT (see Table 5.2). These shells could have easily substituted 1st-Class HT in the PESWEA project. Thus, these shells are worthy of further investigation in future projects. In closing, Westrail and Edith Cowan University believe that expert system technology can now be integrated into the mainstream programming techniques at Westrail.

BIBLIOGRAPHY

Al-Zobaidie A. & Grimson, J.B. (1987). Expert systems and database systems: How can they serve each other?. *Expert Systems, 4* (1), 30-37.

Ansari A. & Modarress B (1990). Commercial use of expert systems in the U.S. *Journal of Systems Management, 41* (12), 10-13.

Barr V. (1990). Exploring expert systems. In P.G. Raeth (Ed.). *Expert systems: A software methodology for modern applications*, (pp. 76-81). Los Alamitos: IEEE Computer Society.

Baroff J., Simon R., Gilman F. & Shneiderman B. (1988). Direct manipulation user interfaces for expert systems. In J.A. Hendler (Ed.). *Expert systems: The user interface*, (pp.99-126). Norwood: Ablex Publishing.

Benchimol G., Levine P. & Pomerol J.C. (1987). *Developing expert systems.* London: North Oxford.

Bielawski L. & Lewand R. (1988). *Expert systems development: Building PC-based applications.* Wellesley: QED Information Sciences.

Bowerman R.G. & Glover D.E. (1988). *Putting expert systems into practice.* New York: Van Nostrand Reinhold.

Breuker J. & Wielinga (1987). Use of models in the interpretation of verbal data. In A.L. Kidd (Ed.). *Knowledge acquisition for expert systems: A practical handbook*, (pp. 17-43). New York: Plenum Press.

Brownstein I. & Lerner N.B. (1982). *Guidelines for evaluating and selecting software packages*. New York: Elsevier Science.

Buchanan B.G. & Smith R.G. (1989). Fundamentals of expert systems. In A. Barr, P. Cohen & E. Feigenbaum (Eds.). *The handbook of artificial intelligence volume IV*, (pp. 149-192). Reading: Addison-Wesley.

Citrenbaum R., Geissman J. & Schultz R. (1990). Selecting a shell. In P.G. Raeth (Ed.). *Expert systems: A software methodology for modern applications*, (pp. 52-61). Los Alamitos: IEEE Computer Society.

Coleman T. (1989). *Expert systems for the data processing professional.* Manchester: NCC Publications.

Collins H.M. (1990). *Artificial experts: Social knowledge and intelligent machines*. London: MIT Press.

Crofts A.E., Ciesielski V.B., Jenkins W., Molesworth M., Smith T., Lee R. (1989). Bridging the gap between prototype and commercial expert systems - a case study. In J.R. Quinlan (Ed.). *Applications of Expert Systems Volume 2.*, (pp.93-105) Sydney: Addison-Wesley.

Dologite D. (1987). Developing a knowledge-based system on a personal computer using an expert system shell. *Journal of Systems Management, 38* (10), 30-37.

Fehsenfeld B. (1988). Application of expert systems: Perspectives from industry. In T. Bernold & U. Hillenkamp (Eds.). *Expert systems in production and services: Impact on qualifications and working life.* Amsterdam: Elsevier Science.

Francioni J. & Kandel A (1988). A software engineering tool for expert system design. *IEEE Expert, 3* (1), 33-41.

Freedman R. (1990). 27-Product wrap-up: Evaluating shells. In P.G. Raeth (Ed.). *Expert systems: A software methodology for modern applications,* (pp. 70-75). Los Alamitos: IEEE Computer Society.

Gevarter W.B. (1990a). The basic principles of expert systems. In P.G. Raeth (Ed.). *Expert systems: A software methodology for modern applications,* (pp. 17-32). Los Alamitos: IEEE Computer Society.

Gevarter W.B. (1990b). The nature and evaluation of commercial expert system building tools. In P.G. Raeth (Ed.). *Expert systems: A software methodology for modern applications,* (pp. 34-51). Los Alamitos: IEEE Computer Society.

Gillies A.C. (1991). *The integration of expert systems into mainstream software.* London: Chapman & Hall.

Guida G. & Tasso C. (1989). Building expert systems: From life cycle to development methodology. In G. Guida & C. Tasso (Eds). *Topics in expert system design: Methodologies and tools,* (pp. 3-23). Amsterdam: Elsevier Science.

Harmon P. & King D. (1985). *Artificial intelligence in business*. New York: John Wiley & Sons.

Harmon P., Maus R., Morrissey W. (1988). *Expert systems: Tools and applications*. New York: John Wiley & Sons.

Hart A. (1989). *Knowledge acquisition for expert systems*. London: Kogan Page.

Hayes-Roth F., Waterman D., Lenat D. (1983). *Building expert systems*. Reading: Addison-Wesley.

Hendler J. & Lewis C. (1988). Introduction: Designing interfaces for expert systems. In J.A. Hendler (Ed.). *Expert systems: The user interface*, (pp.1-14). Norwood: Ablex Publishing.

Hoffman R. (1987). The problem of extracting the knowledge of experts from the perspective of experimental psychology, in *AI Magazine*, Summer 1987.

Howard H. & Rehak D. (1989). KADBASE: Interfacing expert systems with databases. *IEEE Expert*, *4* (3), 65-76.

Jarke M. & Vassiliou Y. (1984). Coupling expert systems with database management systems. In W. Reitman (Ed.). *Artificial intelligence applications for business*, (pp. 65-85). Norwood: Ablex Publishing.

Jongeling S. & Peel G. (Eds.) (1991). *Referencing guide*. Perth: Edith Cowan University, Division of Academic Programmes.

Keim R.T. & Jacobs S. (1986). Expert systems: The DSS of the future?. *Journal of Systems Management, 37* (12), 6-14.

Kinnucan P. (1988). Computers that think like experts. In A. Gupta & Prasad B.E. (Eds.). *Microcomputer-based expert systems,* (pp. 16-26). New York: IEEE Press.

Naughton M.J. (1989). A specific knowledge acquisition methodology for expert systems development. In J. Liebowitz & D.A. DeSalvo (Eds.). *Structuring expert systems,* (pp. 25-52). Englewood Cliffs: Yourdon.

Martin J. & Oxman S. (1988). *Building expert systems: A tutorial.* New Jersey: Prentice-Hall.

McGraw K.L. & Harbison-Briggs K. (1989). *Knowledge acquisition: Principles and guidelines.* Englewood Cliffs: Prentice-Hall.

Merritt D. (1989). *Building expert systems in Prolog.* New York: Springer-Verlag.

Myers W. (1990). Introduction to expert systems. In P.G. Raeth (Ed.). *Expert systems: A software methodology for modern applications,* (pp. 7-16). Los Alamitos: IEEE Computer Society.

Parsaye K. & Chignell M. (1988). *Expert systems for experts.* New York: John Wiley & Sons.

Philip G. (1991). A case study in selecting an expert system shell. *Journal of Systems Management, 42* (1), 32-35.

Plant L., Smalley R. & Waterson D. (1990). EESI: A case study of a successful knowledge base systems pilot. In J. Longwood (Ed.), *1990 Australian Joint Conference on Artificial Intelligence: Workshop Proceedings*, (no page numbers). Sydney: Australian Computer Society.

Rada R. (1989). Building expert databases. In J. Liebowitz & D.A. DeSalvo (Eds.). *Structuring expert systems*, (pp. 145-170). Englewood Cliffs: Yourdon.

Raeth P.G. (1990). Two PC-based expert system shells for the first-time developer. In P.G. Raeth (Ed.). *Expert systems: A software methodology for modern applications*, (pp. 82-89). Los Alamitos: IEEE Computer Society.

Ramsey C.L. & Schultz A.C. (1989). Knowledge representation methodologies for expert systems development. In J. Liebowitz & D.A. DeSalvo (Eds.). *Structuring expert systems*, (pp. 273-326). Englewood Cliffs: Yourdon.

Reichgelt H. & van Harmelen F. (1985). Relevant criteria for choosing an inference engine in expert systems. In M. Merry (Ed.). *Expert Systems 85*, (pp. 21-30). Cambridge: University of Cambridge.

Richer M.H. (1986). An evaluation of expert system development tools. *Expert Systems, 3* (3), 166-183.

Roberts T. (1990). *The development of an expert system for the diagnosis of diseases in fibre and goats*. Unpublished masters dissertation. WACAE, Perth.

Rothenberg J. (1989). Expert system tool evaluation. In G. Guida & C. Tasso (Eds.). *Topics in expert system design: Methodologies and tools*, (pp. 205-229). Amsterdam: Elsevier Science.

Scott A.C., Clayton J.E. & Gibson E.L. (1991). *A practical guide to knowledge acquisition*. Reading: Addison-Wesley.

Stelzner M. & Williams M.D. (1988). The evolution of interface requirements for expert systems. In J.A. Hendler (Ed.). *Expert systems: The user interface*, (pp.285-306). Norwood: Ablex Publishing.

Thomas W. & Hapgood W. (1989). *Reference Manual*. 1st-Class Expert Systems Inc.

Timekeeping Correspondence Course: Railway Employees Award, Marketing and Operating (1988). Perth: Westrail.

Townsend C. (1987). *Mastering expert systems with Turbo Prolog*. Indianapolis: Howard W. Sams & Co.

Williams C. (1990). Expert systems, knowledge engineering, and AI tools - an overview. In P.G. Raeth (Ed.). *Expert systems: A software methodology for modern applications*, (pp. 2-6). Los Alamitos: IEEE Computer Society.

APPENDIX A

THE SECTIONS OF THE RAILWAY EMPLOYEES AWARD IMPLEMENTED INTO PESWEA

GUARANTEED WEEK

1. Each employee other than a casual will be entitled to a full weeks work between Monday and Saturday of 40 hours.

2. However, the employer shall be entitled to deduct payment for any day, or portion of a day, upon which a worker cannot be usefully employed because of any strike or to deduct payment for any day upon which a worker cannot be usefully employed for any cause beyond their control whereby they find himselves unable to carry on either wholly or partially the complete running of trains, services, workshops or other normal operations.

3. Provided that a worker, who cannot be usefully employed because of any strike and who is required for duty on any day and does so report shall be paid a minimum of four hours pay at ordinary rates.

4. A worker who is stood down in accordance with paragraph 2 may elect to be paid for any day but in such case their annual leave entitlement shall be reduced accordingly.

5. A worker stood down in accordance with paragraph 2, shall not lose any sick leave credit or other rights or privileges to which such worker would ordinarily be entitled under this award provided they resume duty within a reasonable time of being so required after such stand down and provided further than this provision does not entitle a worker to payment for any holiday occurring during such period of stand down.

6. The provisions of paragraphs 2, 3, 4 and 5 shall not apply to any worker who is working away from their home station or depot until they are returned to that station or depot or unless the employer and the union concerned agree otherwise.

7. Each week shall stand by itself.

8. The guaranteed period may also be reduced as follows:-

   (a) When an employee is under suspension from duty on account of misconduct. Provided that any worker suspended on a charge which is not sustained shall be entitled to the benefit of the guarantee during the period of suspension.

   (b) When an employee loses time for their own convenience.

9. The following notes and examples are set out in amplification of the previous paragraphs:-

(a) Time worked on Sunday does not count in the week's work of 40 hours. For instance if an employee (Traffic Section) worked from 2100 hours Sunday to 0500 hours Monday, the 5 hours from 0001 hours to 0500 hours Monday would form part of the 40 hours constituting the week's work. On the other hand, if the employee worked from 2100 hours on Saturday to 0500 hours Sunday, the 3 hours from 2100 hours to midnight, would form part of the 40 hours constituting the week's work.

(b) Overtime allowances must not be used to make up time to the guaranteed number of hours for the week. When time is to be made up to 40 hours, the time so made up is to be shown in column of timesheet headed "Time Added - Guaranteed Week".

(c) Time to be made up to 40 hours for the week should, in all cases, be paid at the lowest rate of pay at which time worked in the particular week is being paid for.

(d) Travelling time must be included with the week's work to satisfy the guarantee of 40 hours.

(e) Held away from Home Allowance which falls between 0001 hours Monday and 2400 hours Saturday may be used to make up time to 40 hours, if necessary.

(f) If the time for the week is less than 40 hours, it must be made up to 40 hours, provided the employee did not lose time as per paragraphs 8(a), or (b).

(g) The guaranteed weeks work will be worked over 5 shifts in a week between Monday and Saturday inclusive.

(h) Time worked or travelling time on the employees rostered day off which attracts the 50% penalty and for Guards only, 100% penalty will not be included as part of the guaranteed week's work.

(i) Where a workers rostered day off is altered and an alternative day substituted later in the weeks, time worked on the original rostered day off forms part of the guaranteed weeks work even though it may attract a 50% penalty.

EXAMPLE - Operations Assistant - Traffic Section.

M.	T.	W.	Th.	F.	S.	Hrs	Time Added to make 40 hours	Total for week
8	7	8	8	8	-	39	1	40
6	10 *	6	8	8	-	38	2	40

* Daily O/T Allowance = 1 hour and cannot be used to make up short time.

10. In the case of a Guard or a worker booked to assist the guard on a train all time paid at double time for time worked in excess of 11 hours in a shift shall stand alone and be paid for in addition to the weeks work.

11. For a Guard or a worker booked to assist a guard , time up to and including 11 hours only can be counted towards the guaranteed week.

EXAMPLE - Guard

Sun.	M.	T.	W.	Th.	F.	Sat.	Hours
	6	6	12	8	8		40 Worked
			*				1 Make up
						Total	41

Only 11 hours count towards the guaranteed week and therefore 1 hour make up time is paid.

HOURS OF DUTY

1. The weeks work of 40 hours is to be arranged over 5 shifts Monday to Saturday inclusive with a rostered day off shown on the roster.

2. All employees other than Office Cleaners are paid under the 38 hour week/19 day cycle between Monday and Saturday. Part time Caretakers and Attendants are not covered by the Award and do not work 40 hours per week.

3. The method of payment of the 38 hour week/19 day cycle is that the staff work a basic 80 hours in one pay period and 72 hours in the other pay period totalling 152 hours for the four weekly cycle. However they will receive equal pay for each pay period providing no time is lost without pay or overtime is worked.

4. The average payments for each pay period under the 38 hour week/19 day cycle will be achieved by holding a credit of 4 hours (2 hours a week) from one pay period and including it with payment for the other pay period. See examples on the following pages.

5. The week in which the "Credit Day" occurs the employee shall be guaranteed 32 hours.

6. When the average credit time (2 hours on weekly timesheet and 4 hours on fortnightly timesheet) is to be deducted on the timesheet in the pay period which does not include a "Credit Day" it must be circled in red ink, ie. (4) CR or 2 (CR) as the case may be.

7. Where a worker works a continuous shift Sunday into Monday, such shift, unless it extends into four hours on Monday will not be counted as one of the five week day shifts.

8. The rostered day off must be clearly indicated on the workers timesheet.

OVERTIME ALLOWANCE

1. For all workers covered under the definition of Traffic Section, other than Watchmen, Waiting Room Attendants and Office Cleaners, all time worked in excess of 8 hours in any one of the rostered shifts for the week shall be paid as under:-

   First 3 hours - Time and a half - 50%
   Thereafter - Double Time - 100%

   Provided that in the case of a Guard or a worker booked to assist the Guard on a train, all time paid at the rate of double time i.e. time in excess of 11 hours, shall stand alone and be paid for in addition to the week's work.

2. All time, (exclusive of Sunday time and time worked on the rostered day off paid at time and a half or in the case of Guards double time worked on rostered day off) worked in excess of 40 hours (32 hours in the week where 32 hours are rostered owing to the clearance of a Credit Day) in any one week is paid at the rate of time & a half (50%).

3. Overtime provided for in paragraphs 1 and 2 shall not be paid for twice, payment shall be calculated on the daily or weekly basis, whichever of these alternatives gives the greater amount to the employee.

4. When the time to be paid to an employee whilst travelling, waiting, or for special allowances, added to the working time exceeds 8 hours in any one shift, or 40 hours for the week, overtime allowance is only to be paid on the working time in excess of the hours mentioned.

5. Time absent on paid leave, also walking time, is to be counted as working time.

6. The examples set out on the following timesheets demonstrate the method of applying the alternative of weekly or daily penalty rates, whichever payment is more favourable to the worker - Students should note that in cases where the daily and weekly overtime rates provide the same result, Daily Overtime shall take precedence, and be shown on the timesheet.

7. Overtime rates shall be computed on the rate applicable to the day on which the time is worked, provided that double time i.e. twice the ordinary rate shall be the maximum payment.

8. Weekly overtime allowance should be paid at the rate applicable to the capacity in which the last shift of a particular week is worked. Travelling time is not treated as working time.

9. Time worked daily must be recorded on the timesheet to the nearest quarter of an hour as follows:-

0	minutes	to	7	minutes	inclusive	- Nil
8	"	to	22	"	"	- 1/4 hour
23	"	to	37	"	"	- 1/2 hour
38	"	to	52	"	"	- 3/4 hour
53	"	to	60	"	"	- 1 hour

10. Daily overtime allowance on shifts worked in excess of 8 hours, Monday to Friday, must be paid in accordance with the scale shown hereunder:-

	HOURS WORKED	DAILY OVERTIME ALLOWANCE	HOURS WORKED	DAILY OVERTIME ALLOWANCE	OVERTIME RATES
(a)	8 1/4	Nil	9 3/4	1	
	8 1/2	1/4	10	1	
	8 3/4	1/2	10 1/4	1	8 to 11 hours
	9	1/2	10 1/2	1 1/4	at 50%
	9 1/4	1/2	10 3/4	1 1/2	
	9 1/2	3/4	11	1 1/2	
(b)	11 1/4	1 3/4	12 3/4	3 1/4	
	11 1/2	2	13	3 1/2	Over 11 hours
	11 3/4	2 1/4	13 1/4	3 3/4	at 100%
	12	2 1/2	13 1/2	4	
	12 1/4	2 3/4	13 3/4	4 1/4	
	12 1/2	3	14	4 1/2	

11. The following table shows the calculation of the weekly overtime penalty of time and a half (50%):-

Hours	50%	Hours	50%	Hours	50%	Hours	50%
1/4	-	1 1/4	1/2	2 1/4	1	3 1/4	1 1/2
1/2	1/4	1 1/2	3/4	2 1/2	1 1/4	3 1/2	1 3/4
3/4	1/4	1 3/4	3/4	2 3/4	1 1/4	3 3/4	1 3/4
1	1/2	2	1	3	1 1/2	4	2

The scale shown in paragraph 10 should be memorised. It will be observed that the daily overtime allowance on shifts of 11 hours or more, may quickly be calculated by deducting 9 1/2 hours.

SATURDAY TIME - SHIFT WORKERS

1. Subject to the maximum payment of double time, all time worked on Saturdays by "shift workers" shall be paid for at the rate of time and a half.

2. A "shift worker" means a worker whose usual hours of duty commence and complete other than during the period 0700 hours to 1730 hours.

3. It must be understood that where an employee has been classified as a "Shift Worker" in accordance with the definition given to Para 2, all time on Saturdays up to the first 8 hours must carry the 50% penalty rate, plus 1/4 hour penalty for each quarter hour (1/4 hour) worked in excess of 8 hours, even though such time worked may have been within the hours 0700 to 1730.

   In effect this means that for the first 8 hours we pay time and a half (50% penalty) and thereafter double time (100% penalty) for classified shift workers. This is shown on the timesheet as 50% to Saturday penalty and the balance to daily overtime at 50% (see example).

4. In connection with the general definition of a shift worker, where an employee is regularly rostered to work outside the hours 0700 and 1730, even though it only be for one day per week, such employee is a "shift worker".

5. If a day worker relieves a "shift worker" for a period of one week or more continuously, and during that period of relief is rostered to work outside the hours 0700 and 1730 on any shift, they are entitled to payment of all time worked on any Saturday falling within that period of relief at the rate of time and a half.

6. In the case of a day worker who is not relieving a shift worker, and who is not regularly rostered to work any shift outside the hours of 0700 and 1730 but who is occasionally rostered outside those hours, each fortnightly pay period is to stand alone, and where a majority of shifts in that fortnight are rostered outside 0700 hours to 1730 hours the time worked on both Saturdays will be paid at time and a half. Where 50% or less of the shifts in the fortnight are rostered outside the hours of 0700 and 1730 shift work rates will not apply to either Saturday.

7. In the case of a worker who is recognised as a "shift worker" and is called upon to relieve a day worker for a period of one week or more continuously, they should not be paid the shift work penalty rates on the time worked on any Saturday during the period of relief. If they relieve for a period of less than one week, the worker shall not be treated as a day worker.

8. Area Train Despatchers, Guards, Ticket Examiners on trains, Conductors (Train), Ticket Examiners (Suburban), and Motor Bus Drivers are, with a few exceptions, always regarded as shift workers and accordingly no specific mention need be made of that fact on their timesheets. Timesheets for other workers have the words "Shift Worker/Not a shift worker" printed on them and it is the responsibility of Officers rendering timesheets to see that whichever is not applicable is deleted from the timesheet.

9. The 50% and 100% penalty for Saturday work must be entered on timesheets of employees on the lines provided. In all cases, where Saturday time is worked, the commencing and finishing times of the shift must be specially shown.

10. Provided the worker is entitled to overtime and that the maximum of double time is not exceeded, overtime allowance applicable to time worked on Saturday must be paid in addition to the 50% penalty for shift work.

11. If an Operations Assistant, who is a shift worker, worked 12 hours on a Saturday and the shift was the fifth worked in that week, they would be paid as under:-

Hours Worked	Penalties		
	Saturday Penalty	50% Daily Overtime	100% Daily Overtime
8	4		
3	1 1/2	1 1/2	
1	1/2		1/2 *
12	6	1 1/2	1/2

Total hours payable for shift is 20

* Under normal circumstances 1 hour would be paid but as 1/2 hour is already paid as Saturday penalty the maximum of double time (100% penalty) prevails.

12. Their timesheet would therefore show 12 hours worked on Saturday, 6 hours Saturday penalty, 1 1/2 hours daily overtime at 50% and 1/2 hour daily overtime at 100% giving a total of 20 hours for the shift.

SATURDAY TIME - OTHER THAN SHIFT WORKERS

13. All workers employed after 1230 hours on Saturdays shall be paid for all time worked on that day (prior to and after 1230 hours) under the same conditions as a shift worker (see para 3 and the example).

14. Any worker who is not classified as a shift worker and does not work after 1230 hours on a Saturday is paid ordinary time only.

15. As pointed out in paragraph 9, it is important that the commencing and finishing times of Saturday shifts should be shown on the timesheet.

7530-089-666

WESTRAIL—WAGES TIMESHEET

EXAMPLE : SATURDAY TIME FOR A SHIFT WORKER

TRAFFIC BRANCH

PAY PERIOD ENDING ..... 19 .....

NAME 57

SERVICE No.

DESIGNATION GUARD

RATE OF PAY $

RELIEVING AT

SINCE

HOME STATION

WAGES PAYABLE AT

Accounts Branch Use Only	Final Payment	Extend Paid Up to Date to	Deductions No. of Periods	Deductions Comm. Period	Taxation No. of Periods

TIMES MUST BE CLEARLY SHOWN DAILY

	First Week Ending Sun	Mon	Tue	Wed	Thur	Fri	Sat	Second Week Ending Sun	Mon	Tue	Wed	Thur	Fri	Sat
ON DUTY		0600	0600		0600	0600	0700		1400	1400		1130	1300	0700
MEAL														
OFF DUTY		1400	1400		1400	1400	1600		2200	2100		1600	2100	1630
SHIFT WORK ALLOWANCE	XXXXX						XXXXX	XXXXX						XXXXX

* SHIFT WORKER

NOT A SHIFT WORKER

N.B. A Shift Worker is one whose usual hours of duty commence and complete other than during the period 07.00 to 17.30

HOW EMPLOYED — DESCRIPTION OF WORK	ACTING Designation	ACTING Date	Period No.	Class of Labour	Penalty Code	ACCOUNT OR WORK ORDER	HOURS WORKED — FIRST WEEK (IN HOURS PER DAY) Sun	Mon	Tue	Wed	Thur	Fri	Sat	SECOND WEEK (IN HOURS PER DAY) Sun	Mon	Tue	Wed	Thur	Fri	Sat	* TOTAL HOURS Units	Dec	† ACTING Designation/Date
AVERAGING ADJUSTMENT				16										(4)							4	00	CR
								8	8		8	8	8		8	8		6½	8	9½	80	00	
										RESTED DAY OFF							RESTED DAY OFF						
* PAID LEAVE PH — OFF DUTY				90																			
* SICK				94																			
* SICK ARREARS				94																			
* ANNUAL				92																			
* LONG SERVICE				98																			
* DAILY TOTAL — EXCLUDING PENALTIES								8	8		8	8	8		8	8		6½	8	9½			
TOTAL ALLOWED (AVERAGED AT 40 P.W.)				77																			
TOTAL HOURS EACH WEEK — EXCLUDING PENALTIES								40						(4)	40						76	00	
* PENALTY TIME (DAILY ADJUSTMENT)					24															¾		75	
* DAILY ADJUSTMENT					25																		
* WEEKLY ADJUSTMENT					33																		
* SATURDAY					44								4							4¾	8	75	
* SICK LEAVE PENALTY (SATURDAY)				94	44																		
* SUNDAY					55																		
* 6TH SHIFT					61																		
* ADJUSTING DAY					37																		
* PH PENALTY					47																		
* PH PAYMENT					91																		
TOTAL HOURS — INCLUDING PENALTIES								44						(4)	45½						85	50	
* NON PAID HOURS																							
* INITIALS OF TIMEKEEPER																							

ALLOWANCES			Allowance Code			Units	Dec	Amount £	c
* EXPENSES			100						
* SHIFT WORK				FIRST 8 HOURS AT 50%					
* SHIFT WORK — SICK LEAVE		94		(4 HOURS) SATURDAY					
				PENALTY.					
* DISTRICT ALLOWANCE									
* COMPENSATION CLAIM No.		87							
TOTAL ALLOWANCES									

OVER 8 HOURS AT 100% PENALTY 150%

AT ¾ HOUR SATURDAY

PENALTY AND ¾ HOUR DAILY

OVERTIME

INITIALS OF ACCOUNTS BRANCH OFFICERS

LEAVE CHECKED AND ENTERED

TIMEKEEPING CHECKED

APPROPRIATED

LEAVE DETAILS

SIGNATURE OF EMPLOYEE

SIGNATURE OF OFFICER IN CHARGE

/ /19

DISTRICT ALLOWANCE. EMPLOYEES TO SUPPLY DETAILS AS PER CLAUSE 32(4) OF AWARD.

DISTRICT ALLOWANCE CLAIMED . . . P.W.

STATE MARRIED OR SINGLE

ARE DEPENDANTS LIVING WITH CLAIMANT AT HOME STATION?

* TO BE COMPLETED FOR BOOKINGS OUTSIDE THE PERIOD ONLY

113023/7/83 1400 Pds 1 3513