On Case Representation and Indexing in a Case-Based Reasoning System for Waste Management

Karl Lyndon Wortmann, BSc(Hons)

submitted in fulfilment of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

in the

Department of Computer Science and Information Systems
University of Natal

1997

ABSTRACT

Case-Based Reasoning is a fairly new Artificial Intelligence technique which makes use of past experience as the basis for solving new problems. Typically, a case-based reasoning system stores actual past problems and solutions in memory as cases. Due to its ability to reason from actual experience and to save solved problems and thus learn automatically, case-based reasoning has been found to be applicable to domains for which techniques such as rule-based reasoning have traditionally not been well-suited, such as experience-rich, unstructured domains. This applicability has led to it becoming a viable new artificial intelligence topic from both a research and application perspective.

This dissertation concentrates on researching and implementing indexing techniques for case-based reasoning. Case representation is researched as a requirement for implementation of indexing techniques, and pre-transportation decision making for hazardous waste handling is used as the domain for applying and testing the techniques.

The field of case-based reasoning was covered in general. Case representation and indexing were researched in detail. A single case representation scheme was designed and implemented. Five indexing techniques were designed, implemented and tested. Their effectiveness is assessed in relation to each other, to other reasoners and implications for their use as the basis for a case-based reasoning intelligent decision support system for pre-transportation decision making for hazardous waste handling are briefly assessed.

PREFACE

The experimental work described in this dissertation was carried out in the Department of Computer Science and Information Systems, University of Natal, Pietermaritzburg, from January 1996 to December 1997 under the supervision of Dr D. Petkov of Department of Computer Science and Information Systems, University of Natal, and Professor E. Senior, Director of the International Centre for Waste Technology (Africa).

These studies represent original work by the author and have not otherwise been submitted in any form for any degree or diploma to any University. Where use has been made of the work of others it has been duly acknowledged in the text.

ACKNOWLEDGEMENTS

I would like to thank both my supervisor, Dr. D. Petkov, and co-supervisor, Prof. E. Senior, for the privilege of being able to study under them. Thank you to Dr. D. Petkov for always being available to give advice, for encouraging me, for providing his expertise and spending untold hours proof-reading and assisting me with my work. Thank you to Prof. E. Senior for sound advice, a wealth of knowledge on hazardous waste, and the opportunity to apply my computer science research to a worthwhile real-world problem.

Thank you to Waste-tech (Pty) Ltd for providing the data necessary for my research. Thanks in particular to Mary Chettle, Dawn Flay and Melanie Francis for giving up their valuable time to explain the handling of hazardous waste at Waste-tech, and providing their expertise to assist me in the designing of my techniques.

The financial support of the FRD is gratefully acknowledged.

Thanks to my family for all the love you have given me before and during my research.

Thank you to my wife Joanne for your love and support, which helped me through the difficult parts of this work. Thank you for always being there.

CONTENTS

INTRODUCTION

Case-Based Reasoning (CBR) is an Artificial Intelligence (AI) technique which solves a problem by retrieving stored past problems and solutions (cases) which are relevant to the current problem and reusing their solutions to solve the new problem (Leake, 1996; Aamodt & Plaza, 1994). Literature (eg Selfridge & Cuthill, 1989; Slade, 1991; Barletta, 1991; Aamodt & Plaza, 1994; Leake, 1996) generally credits work by Roger Schank as the first to describe the CBR approach to reasoning. As a technique which reasons by means of stored past experiences, it has demonstrated applicability to areas previously unsatisfactorily handled by traditional techniques such as Rule-Based Reasoning (RBR) (Kolodner, 1993; Gupta, 1994). Demonstration of its successful applicability has led to it becoming a viable new AI topic from both a research and application perspective.

The basic premise of CBR is that in order to solve or help solve a new problem, past problems and their solutions which contain information relevant to the new problem are retrieved from storage, and presented for solution construction (Barletta, 1991). In order to allow such reasoning, the basic structure of a CBR system consists of a knowledge base in which past problems and solutions are stored, along with procedures for retrieving relevant problems and solutions (Gupta, 1994). This knowledge base is generally referred to in the literature as Case Memory, while each encoded problem description and solution is referred to as a Case.

In order to allow successful reasoning from this case memory, the following issues must generally be addressed when developing a CBR system :

Case Representation	-	How an actual real-world case will be represented in the system.
Indexing	-	How cases in case memory will be labelled so as to ensure the correct cases are found at retrieval time.

Case Retrieval	-	What algorithms should be employed in order to most efficiently search and match case memory.
Adaptation and Solution Construction	-	How actual solutions are produced based on possibly inexact matches between stored cases and the current problem.

While the first three issues invariably must be addressed in one form or another in a reasoner, the fourth issue, adaptation and solution construction, is often omitted from real world reasoners due to the complexity and inherent risks involved in the task. As a result, CBR systems can be found performing a decision support role rather than as a pure expert system, as suggested in Kolodner (1991). There are however many examples of reasoners which perform adaptation (eg. (Hennessy & Hinkle, 1992); PERSUADER (Sycara, 1991; Sycara, 1993); CHEF (Hammond, 1989)

Environmental management is an issue of global importance which has proven to be a fruitful application area for computer science and information systems (Radermacher, Riekert, Page & Hilty, 1994). Nationally, hazardous waste management has been singled out as an area in need of particular attention (Department of Environment Affairs, 1992). At a national, regional and corporate level, all aspects of hazardous waste handling are in need of attention. Even where the handling cannot necessarily be viewed as wrong or inadequate, areas which could be improved through the application of information technology can still be found. One such a situation is the application of CBR to decision making in the pre-transportation handling of hazardous waste at Waste-tech (SA), which is domain for the testing of techniques implemented in this research.

Research Issues and Objectives

The main aim of the dissertation was to research indexing in CBR, and to implement and test a number of indexing techniques. In order to enable implementing and testing of techniques, it was required that case representation also be researched. To test techniques, a test domain was chosen where use of CBR was deemed applicable.

As such, the specific sub-objectives were to :

- Research CBR in general in order to place research on indexing in context.
- Research Case Representation issues to facilitate the design of an accurate case representation in order for indexing to be successfully implemented and tested.
- Research Indexing techniques as the main focus of the dissertation.
- Choose a test domain where CBR is applicable to decision support in the domain, and determine how CBR could be applied to the domain.
- Design a number of indexing techniques which are:
  - Valid based on CBR indexing theory.
  - Applicable to the test domain.
- Test these techniques to determine:
  - Which performs best from an indexing perspective.
  - Whether performance is satisfactory compared to other CBR implementations performance.

Scope and Delimitations of the Research

The research concentrated on indexing of cases. However, in order to implement indexing techniques, a satisfactory case representation is required. As such, the two areas of CBR which receive specific attention were:

Case representation - This was researched in order to allow the design of a valid case representation to enable implementation and testing of indexing techniques. A single case representation was designed.

Indexing techniques - As the major focus of the research, indexing was researched in detail. Five indexing techniques were designed, implemented and tested.

A review of CBR was given. Here issues which are outside the focus of the research, but still of relevance to CBR (eg case adaptation) are discussed briefly in order to give a balanced view of CBR.

For retrieval of cases, the nearest neighbour approach (Barletta, 1991) was followed in all experiments. The indexing techniques implemented focussed on the identification of relevant knowledge at the case level, not on organising cases in memory, thus the nearest neighbour approach was deemed most applicable for search.

As it was not the focus of the research, adaptation of cases is not investigated in detail, or implemented. The proposed role of a system implemented from the techniques designed and tested in this dissertation would be a decision support one as discussed in Kolodner (1991), whereby retrieved cases are presented to assist the human expert in decision making.

As the focus of the dissertation is on indexing in CBR, no complete decision support system was built. Instead, the techniques designed and software written were implemented with the possibility of a decision support system arising from the techniques in mind.

Significance of the Research

The significance of the research can be measured in two areas. Firstly, CBR papers tend to use varied terminology for describing techniques designed. In addition, many principles involved in the design and implementation of CBR techniques tend to remain unmentioned in the literature. As such, research which attempts to identify and describe the principles behind case representation and indexing, as well as investigate and describe their implementation, is potentially of significance to CBR practice.

Secondly, the chosen domain of pre-transportation handling of hazardous waste is one which is rich in instance information (i.e. cases), but is not yet well structured to facilitate decision making for anyone other than the expert currently involved in the process. The application of CBR to the domain provides a useful example of how the current instance knowledge could be used to assist in making the current decision making processes more accurate.

Research Method

Literature on CBR (Journal publications; Conference proceedings; World Wide Web resources) was surveyed to cover:

- A general overview of CBR, covering:
  - The origins of CBR.
  - The major components of CBR.
  - The generalised CBR process.
  - The major research areas in CBR.
- Case representation theory and techniques in detail.
- Indexing theory and techniques in detail.

Literature was also surveyed to assess how computing had been applied to hazardous waste issues. The information from the survey of hazardous waste technology was used primarily to assist in justifying the choice of application domain. Interviews were carried out with Waste-tech experts involved in the current decision making process. This information was used to determine the nature of the current decision making process, assess the need for a decision support system, determine where a decision support system could fit into the process, and obtain the data source for the application of CBR.

For design and implementation of techniques, the methodology followed was the engineering method as described in Glass (1995), which essentially involves observing existing solutions, proposing improvements, building them, and measuring and analysing them.

One case representation was implemented. This was designed based on the principles of case representation as determined from the literature, used a database record structure, and was built primarily with the aim of enabling the implementation of the indexing techniques designed.

Five indexing techniques were implemented based on the principles of indexing. These were :

- Equal Weight Nearest Neighbour (EWNN).

- Fixed Weight Nearest Neighbour using Expert Judgement (FWNN-EXP).
- Fixed Weight Nearest Neighbour using the 80-20 Rule (FWNN-80:20).
- Variable Weight Nearest Neighbour (VWNN).
- Separate Field Variable Weight (SFVW).

These represent an evolution from a single, global indexing technique to a single, case-specific indexing technique to multiple, case-specific indexing. All five techniques were implemented in custom built modules. In addition, EWNN and FWNN-80:20 were implemented in ESTEEM (Watson, 1995), a commercially available CBR package.

Testing techniques used in a number of CBR implementations (see Chapter 6 for references) were used to determine benchmark tests for testing of CBR systems. Using a case base of 90 cases created by the author, the following test results were gathered from the experiments :

- Retrieval time.
- Average matching of retrieved cases.
- Average matching of the top ten or top three cases retrieved, depending on the technique being tested.
- The number of cases retrieved.
- The accuracy of each solution produced (prediction accuracy), compared to the solution as determined by the domain expert.

In the analysis of tests, as the focus of the research was on indexing (which in practice implies the location of the relevant cases at retrieval rime) prediction accuracy was used as the determining factor in assessing the performance of the techniques. While the engineering approach was used for the design and implementation of techniques, the scientific method of comparing results using hypothesis testing as described in Pollard (1977) was used to compare the major result gathered, prediction accuracy. As such, prediction accuracies were compared using paired *t*-tests (as described in Pollard, 1977).

The test results were analysed for four aims:

- The results of custom modules were compared to their counterparts implemented in ESTEEM. This was done to ensure that the custom modules were performing in a comparable manner.
- The results of the various custom modules were compared to each other. This was done to determine which of the indexing techniques implemented performed the best. As such, the prediction accuracies were the main focus.
- The results of the custom modules were compared to results of other real world reasoners. This was done to determine whether or not any of the techniques implemented could be considered a success when compared to other systems implemented.
- Consideration was given as to what implication the results had for possible implementation of a complete system, thus examining whether CBR could be considered a technique applicable to intelligent decision support for decision making for hazardous waste handling.

Overview of the Dissertation Structure

Chapter One gives an overview of case-based reasoning. Chapter Two goes on to discuss case representation in detail, while Chapter Three is a detailed examination of issues relating to indexing. Chapter Four is an overview of the current decision making process at Waste-tech (Pty) Ltd, including a description of how a proposed CBR intelligent decision support system could be integrated into the current decision making process. Chapter Five contains a detailed account and discussion of CBR techniques designed and implemented. Chapter Six gives a description of tests performed, test results and discussion of results. Following this, conclusions are given.

1. AN OVERVIEW OF CASE-BASED REASONING

Case-Based Reasoning (CBR) is an Artificial Intelligence (AI) technique which is based on the use of specific knowledge of past experiences to solve a new problem (Ketler, 1993; Gupta, 1994). In this chapter, a short overview of how CBR arose is given. Following this, the major components of a CBR system are briefly discussed. The generalised CBR process is presented, and a short discussion of each of the research areas in CBR is given. The potential advantages and disadvantages presented by CBR are then examined. Finally, the domains in which CBR is applicable are discussed, and the chapter is concluded.

1.1 The Origins of Case-Based Reasoning

While other authors are also noted as being at the forefront of the development of CBR (Aamodt & Plaza, 1994, for example cited a number of additional sources), literature generally credits the work by Roger Schank on Dynamic Memory in 1982 as the first to describe a case-based type of approach to reasoning (eg Selfridge & Cuthill, 1989; Slade, 1991; Barletta, 1991; Aamodt & Plaza, 1994; Leake, 1996). The literature indicates that the CYRUS system of 1984 by Janet Kolodner was the first system to implement case-based techniques (Slade, 1991; Leake, 1996), and CLAVIER (Hennessy & Hinkle, 1992), an industrial application to plan autoclave loading is indicated by Aamodt & Plaza (1994) to have been the first fielded CBR system.

Following this pioneering of CBR, a number of industrial applications have been built (Allen, 1994), and many CBR publications now appear. CBR papers are published in numerous journals, and a number of conferences devoted specifically to CBR have been organised, eg:

- the DARPA workshops on CBR.
- The European Workshops on Case-Based Reasoning.
- The International Conferences on Case-Based Reasoning.

- The UK Case-Based Reasoning Workshops.
- The German Workshops on Case-Based Reasoning.

Commercial CBR tools have become available, such as REMIND, CBR Express and ESTEEM (Watson, 1995), and CBR applications are now used for a variety of industrial problems such as industrial design, planning, and explanation systems (Thompson, 1997). In addition, a number of World Wide Web (WWW) pages have appeared in recent times with detailed information on CBR developments. Some of the current examples are:

- The AI-CBR Home Page
  `http://www.ai-cbr.org`
- The Case-Based Reasoning Repository
  `http://www-cia.mty.itesm.mx/~lgarrido/Cbr/cbr.html`
- Case-Based Reasoning on the Web
  `http://wwwagr.informatik.uni-kl.de/~lsa/CBR/CBR-Homepage.html`
- Machine Learning and Case-Based Reasoning Home Pages
  `http://www.aic.nrl.navy.mil/~aha/`
- Case-Based Reasoning: A Categorised Bibliography
  `http://www.salford.ac.uk/survey/staff/IWatson/cbrefs.htm`

1.1.1 The Motivation for the Development of Case-Based Reasoning

As an AI technique with both research potential (as indicated by the growing research in the field) and practical application (indicated by the appearance of commercial CBR tools), it would be unsurprising to view CBR as a successful attempt by the AI community to develop a new reasoning method, and certainly, this view would not be incorrect. This view is put forward by Kolodner (1991). Here it is indicated that while Rule Based Reasoning (RBR) has been used to build successful fully automated expert systems in certain domains, it has not been possible to use such traditional techniques successfully in all domains. Specifically, domains requiring creativity or having incomplete knowledge are mentioned as areas of difficulty. Kolodner (1991) put

forward three possible ways to address the problem, namely to drop AI research completely, to continue with the original AI goal of giving the computer the full capabilities of the human reasoner, or to concentrate on the present problem of building better systems. Kolodner (1991) indicated that it is this third approach which led to the development of CBR. As such, there is certainly justification for viewing the initial development of CBR in this way.

However, as many research papers on CBR indicate (eg Owens, 1988; Selfridge & Cuthill, 1989; Barletta, 1991; Helton, 1991; Kolodner, 1991; Slade, 1991; Gonzalez & Laureano-Ortiz, 1992; Chaturverdi, 1993; Gupta, 1994; Hansen, Meservy & Wood, 1994) CBR is a memory-based approach to reasoning which is said to model the manner in which humans reason. As such, it would not be surprising to find its origins based not just in an attempt to find a successful new AI technique, but also in research on how human memory and reasoning works, and this indeed appears to be the case.

Specifically, the literature indicates two major motivations for the development of CBR. Slade (1991) classifies these as scientific - an attempt to understand and model the way human beings think, and technological - the practical application of AI through the building of systems which are able to perform intelligently in fixed domains. Leake (1996) classifies these in terms of the source of the motivation for the research, listing cognitive science (modelling of human behaviour) and AI (building more effective AI systems). Aamodt & Plaza (1994) give more specific details, indicating the original roots of CBR to be lying in research in philosophy & psychology, followed by the more specific research of Schank in 1982, which led to the development of the first CBR systems. In other words, the literature indicates that while CBR in the modern context is a practical AI technique which currently is receiving great attention from both a theoretical and practical perspective it's origins do not lie only in the motivation to build better practical AI systems, but also in an attempt to model human reasoning.

1.1.2 The Initial Concepts behind Case-Based Reasoning

The details of the principles behind the development of CBR theory would alone entail a separate research dissertation, and as such, they will not be discussed here. Instead, an overview based on

literature from some of the principle developers of the theory will be given here to provide a basic summary of the motivation for the development of CBR theory, and to position this work within the research efforts in the field.

Riesbeck and Schank (1989) contended that the conception of human reasoning as represented in Rule-Based AI systems was incorrect, that in fact human reasoning does not operate from a set of rules. Their contention was that rather than reasoning from first-principles (as a Rule-Based AI system would), people use memories of past situations to understand and produce solutions for new ones. There are two key points here. Firstly, the concept of reasoning by means of past experiences. The second is that Riesbeck and Schank (1989) contended that the key to human reasoning was understanding, and linked to this, explanation. They proposed that in order for people to be able to reason (from past experiences), they had to understand the information they received, and in order to understand the information, they had to be able to explain it. This explanation can be viewed as the reasons why the results of an experience occurred, and as will be discussed later, this explanation is an essential element of a CBR system. In other words, without going into detail, the theory was that humans solve new problems by remembering old problems and solutions. They are able to use old problems because they understand them through a process of explaining the important features of the problem.

It does not however suffice to simply point out that human reasoning works by means of finding past experiences which most closely match the current one, and through understanding them being able to use them. Rather, if one accepts this basic theory as a correct evaluation of how humans solve problems, the important issue, certainly from the point of view of AI, is how this type of problem solving works. Schank (1988) and Riesbeck & Schank (1989) focussed on human memory in general rather than specifically problem solving, and discussed this issue under the title of reminding.

Schank (1988) referred to human memory as a dynamic memory, where a dynamic memory is one which is altered by every experience it processes, i.e. it learns. It is contended (in Schank 1988) that this learning is a feature of a good memory system. In addition to learning, it is stated that the

second important feature of a memory system is that it should be able to find episodes stored in it. Clearly, these are essential requirements of any system which relies on using past problems to solve new ones.

For the purposes of our discussion, a number of points made by Schank (1988) and Riesbeck & Schank (1989) about a dynamic memory system which reasons by use of stored past experiences can be summarised as follows:

- Such a system learns from every experience.
- Is capable of finding relevant episodes when needed.
- Is structured in a manner which will allow such finding to occur.

The point is made that understanding a new experience or problem is based on the knowledge currently in memory. As such, a chess expert would be reminded of a famous game when viewing a similar new one, whereas a non-expert would not. The reason simply being that the expert has such knowledge in memory.

Schank (1988) then went on to discuss how, based on this model of human memory, expert systems should be built. The one option was obviously to extract the expertise and compile it in rules (the traditional approach). The alternative was to model the memory of the expert, building a system which would process new experiences based on old stored ones.

Having already defined CBR previously in the dissertation, we have the advantage of immediately noting that this theory was the essence of CBR. As such, we have a simple view of the cognitive basis for CBR.

For the last part of section 1.1, a brief outline of how it was proposed this theory should be implemented is given.

1.1.3 The Initial CBR implementation proposal

In section 1.1.2 it was discussed that understanding of an experience through explanation was viewed as being the key to the reuse of past problems. From an AI point of view, the issue to

address was what memory representation could allow the storage of experiences in a manner which would allow the relevant experiences to be found for future problem solving.

Riesbeck & Schank (1989) put forward scripts as the initial memory structure used for organising episodic memory. Scripts are memory structures which store information about general situations, and are episodic in nature (Slade, 1991). However, scripts are generalised knowledge of situations built up from exposure to a number of experiences, i.e. they are knowledge generalised from a number of specific instances. As has been pointed out, the theory of dynamic memory worked on the principle that specific past experiences were used when dealing with a new one. As such, two important points about scripts were brought up in Schank (1988). Firstly, scripts should be dynamic, not static data structures. Secondly, particular episodes should be stored as part of scripts.

As such, scripts in their basic form were not satisfactory memory structures. Instead, a new memory structure, the Memory Organisation Package (MOP), was proposed by Schank (Riesbeck & Schank, 1989; Kolodner, 1993). Before describing the basic nature of MOPs, an important point made in Kolodner (1993) should be mentioned. We have already discussed that in dynamic memory, past experiences need to be accessed. Kolodner (1993) specifically discussed the issue of indexing (a term central to most CBR literature), stating that people label memory structures in ways which make them accessible at a later stage. More specifically, they are labelled according to their type, and how they differ from other similar structures.

An example used in Schank (1988) and Kolodner (1993) is that of the Legal Seafood Restaurant. Here unlike most other restaurants, you are asked to pay the bill before the food arrives. In this case, the Legal Seafood Restaurant differs in a specific manner, namely when the bill is brought, and could be indexed on this particular feature. Without referring to MOPs, Schank (1988) suggests that this episode would be stored in memory in terms of how it differed from the normal expectations of a restaurant visit, and would thus be located based on its differences from other experiences. This example of indexing on differences is central to the idea of the MOP.

MOPs, as described by Riesbeck & Schank (1989), are used to represent knowledge about a certain event or event class. They would be used to hold information about general classes of information (i.e. general knowledge), as well as organising specific knowledge (i.e. cases) about the classes in a hierarchical structure. If we look again at the example of the visit to the Legal Seafood Restaurant, we could have a MOP for a restaurant visit which would hold general knowledge about restaurant visits, and a more specific MOP for the Legal Seafood Restaurant visit linked to the general restaurant visit MOP.

Riesbeck & Schank (1989) illustrated a MOP with a "going on a trip" example. They state that a MOP "contains a set of *norms* which represent the basic features of the MOP". These norms were listed as actors, events, goals, etc for the MOP. Thus, according to them, a "going on a trip" MOP would contain a human actor travelling to some place to accomplish some goal. Riesbeck & Schank (1989) highlighted the fact that these MOPs are used especially for complex event knowledge representation. They then further made the point that this "event MOP" holds very much the same information as the scripts mentioned earlier, but the nature of the MOP is very different in that instead of having the static, independent structure of the script, the MOP is a dynamic structure which changes with use.

A MOP can have specializations of its general structure, for example (Riesbeck & Schank, 1989), "going on a business trip" is a specialization of the more general "going on a trip" MOP. In the opposite direction, the "going on a trip" MOP is an abstraction of the "going on a business trip" MOP. At the most specific level, a MOP storing information on a particular event is called an instance. It can be seen how the hierarchical structure mentioned earlier comes into existence.

As mentioned earlier, indexing was put forward as a solution to the problem of organising general and specific knowledge in such a manner that the correct knowledge is retrieved at the correct time (and referred to using the term "indexing" by Kolodner (1993)). Riesbeck & Schank (1989) appeared to refer to these indexes for MOPs as links, of which the following are identified :

Abstraction links - These are the links from a MOP to those MOPs which are abstractions of it, eg. the "going on a business trip" MOP has an abstraction link to the "going on a trip" MOP.

Scene links - These are links from a script-type event to subevents of that script. For example, a "going on a plane trip" event would have scenes(sub-events) for buying a ticket, boarding the plane, etc.

Exemplar links - These are links from a MOP to specific instances of that MOP. For example, a number of instances may have been used to derive a MOP. The MOP is then linked to them by exemplar links. Alternatively, the MOP could be linked to prototypical examples of the MOP.

Index links - These are links from a MOP to specialisations of the MOP. Each index link is labelled with an attribute-value pair. It is these pairs that are used to allow efficient accessing of the MOP. The example of "purpose of trip = pleasure" pair is used. The attribute "purpose of trip" is a norm of "going on vacation", thus every specialization of this MOP will inherit the pair as a norm. In order to reach the MOP, an event would have to have this pair. Note that this index link does not differentiate between specialisations of the "going on vacation" MOP, as each specialisation happens. No doubt, other indexes would be used for distinguishing cases.

The importance of these indexes can be seen in Slade (1991). The example discussed there noted that a story of a man's wife never cooking his steak rare enough has similarity to a story of a man whose barber would not cut his hair short enough. While on the surface the two stories hold little similarity, it can be seen that at a lower level, parallels can be drawn. It is with indexing that these

two apparently different stories can have their similarities highlighted in such a manner that they are remembered at the correct time, albeit that the indexing and organisation of memory would be a highly complex task.

Failure links - These are links from a MOP to instances where the actual instance events are not what the MOP predicted. It is important to note here that the actual instances are not necessarily failures (i.e. goal failures), but only unexpected situations. These could be unexpected successes or failures, from a goal point of view. It is from a reasoning point of view that they are failures.

Having briefly discussed the theoretical basis for the development of CBR from both a cognitive and AI perspective, two points should still be made. Firstly, Schank's research into reminding did not only look at intra-domain analogy, such as one restaurant visit reminding one of another. Rather, attention was also given to cross-domain reminding, such as that presented in the discussion of MOPs index links. While this type of reminding does receive interest in the CBR community, CBR in general can be considered more of an intra-domain analogical technique (Aamodt & Plaza, 1994).

Secondly, in the text so far, the term learning has been used in relation to CBR. It should be noted that the learning being referred to is a learning particular to CBR. Kolodner (1993) points out that in artificial intelligence, learning generally refers to the learning of generalizations. In CBR, while the learning of generalizations is certainly a component of many reasoners, it is not the standard form of learning which CBR systems undertake. Kolodner (1993) states that CBR achieves learning through the accumulation of new cases, and the assignment of indexes (these issues are discussed more specifically in sections 1.2, 1.3 and 1.4). In other words, learning in CBR refers to the ability to obtain new instance (rather than general) knowledge (and hence obtain greater coverage of the problem domain) through accumulation of cases, and improve recall ability by correctly indexing new cases.

Having presented this short discussion on the origins of CBR, the following sections present a short overview of the major issues pertaining to CBR research. Note that, as mentioned earlier, the purpose of these sections is to present an overview of CBR. As they are not the focus of this dissertation, detailed discussions are not presented. In addition, two points should be noted about the sections following. The first is that while certain classifications and terminologies are used in the text, this should not be taken to indicate that these are standards which will be found in all sources. Terminology varies to some degree in the literature, as does classification. In addition, issues are discussed under different headings in different sources. As such, no one classification or set of terms can be viewed as the single correct one.

1.2 Components of a Case-Based Reasoning System

The first section to be discussed is the various components which make up a CBR system. This will be presented in the next four sections, namely Case Memory, Inferencing Mechanism, Modification Mechanism and Indexing Mechanism. As just mentioned, this should not be taken to indicate that all reasoners have all four components, nor that all literature utilises the terminology used here. Rather, in many reasoners, not all these components are present, and, for different reasoners, different components are emphasized more than others. As such, the components presented here represent an amalgamation of information from a number of sources, eg Barletta (1991), Slade (1991), Kolodner (1991), Kolodner & Mark (1992), Ketler (1993) and Gupta (1994).

In addition, these components must also be discussed in reference to the major research areas of CBR, namely case representation, indexing, case retrieval, and adaptation and solution construction. As such, reference will be made to these research areas in the proceeding sections.

1.2.1 Case Memory

In CBR, the case memory, as the name implies, is the repository for the reasoner's cases (often referred to as the Case Base). In a simplified form, case memory could be equated to one or another form of database store in which the cases are kept, one database record for each case. The

memory system used in the TFM system of Bayles & Das (1994) uses such a method for case storage. However, there is great variability in storage methods.

Considerations of case memory generally fall into two areas, namely case representation and memory organisation.

Case representation refers to the method used for representing individual records in case memory. For example, the TFM system (Bayles & Das, 1994) uses flat frame-like structures for representation, as does the research of Petrak, Trappl and Furnkranz (1994), while GRAND (Oosthuizen, 1994) uses a lattice to represent cases and organise case memory. Issues relating to case representation are discussed under the research area of case representation (section 1.4.1), and in more detail in Chapter 2 since they are closely related to the primary scope of this research.

Memory organisation refers to the method used for arranging cases in memory. In reasoners where case memory is organised in some manner other than a simple flat memory (eg hierarchies or discrimination networks), this is generally done either to facilitate accuracy of search for cases (eg CHEF (Hammond, 1989; Kolodner, 1993)), speed of search for cases, (eg GRAND (Oosthuizen, 1994)) or possibly both (eg Battle Planner (Goodman, 1989)). Memory organisation tends to be strongly linked to indexing, and as such is discussed under the area of indexing (section 1.4.2) and in more detail in Chapter 3 where indexing is examined in detail.

1.2.2 Inferencing Mechanism

When an input problem is presented to a reasoner, the reasoner searches case memory for cases which have some applicability to the input problem. It is the responsibility of the inferencing mechanism to ensure that those cases which store information which might be relevant to the input problem will be found by this search. In order to do this, a reasoner should be able to correctly extract the important features of the input problem, and then use those features to search case memory for stored cases which have relevance the input problem.

As with case memory, this component relates to a number of CBR issues, these being :

Knowledge extraction	-	The extraction of important features of the input problem which are used for searching case memory for relevant cases.
Case indexing	-	The labelling of cases to identify those features of a particular case considered useful to later reasoning.
Retrieval algorithms	-	Algorithms used to search case memory.

The integrated nature of CBR can be seen here. For instance, knowledge extraction could be viewed as an indexing issue (eg JUDGE (Riesbeck & Schank, 1989), where input problems are processed to extract information prior to retrieval). Case indexing could be at a case level, and hence unrelated to memory organisation (eg the TFM system (Bayles & Das, 1994), or any other system which uses nearest neighbour search with field weighting), or it could involve construction of some type of hierarchy (eg CHEF (Hammond, 1989)) or network (eg the lattice of GRAND (Oosthuizen, 1994), or the discrimination net of Battle Planner (Goodman, 1989)), and hence involve memory organisation. For the purposes of this dissertation, the knowledge content of a case is discussed under case representation (section 1.4.1 and Chapter 2), case indexing is discussed under its own heading (section 1.4.2 and Chapter 3), while retrieval is discussed briefly under case retrieval (section 1.4.3).

1.2.3 Modification Mechanism

Once cases have been retrieved from case memory, they must be utilised to provide a solution to the input problem. Provided that the reasoner has searched case memory correctly, and provided that case memory contains cases which have some relevance to the input problem, the information for this solution is contained in the retrieved cases. Ideally in CBR, an exact match would be found between the input problem and a stored case. In such a situation, the stored case's solution could be transferred as is to the input problem. However, whenever knowledge is incomplete, no exact match will be found. In such a situation, a solution to the new problem must be constructed from the retrieved case(s). In certain reasoners (eg CLAVIER (Hennessy & Hinkle, 1992; CHEF (Hammond, 1989)), this is achieved by adapting the retrieved cases to match the input case. In

others (eg the work of Petrak, Trappl and Furnkranz (1994)), this modification step is not performed by the reasoner. Instead, cases are presented to the user to assist in decision making.

As adaptation of cases is not the focus of this dissertation, there is no chapter in it devoted to this issue. However, a brief discussion is presented under the research area of adaptation and solution construction (section 1.4.4).

1.2.4 Indexing Mechanism

Once a solution to the input problem has been produced, it is evaluated to ensure validity and/or find errors. When the solution is deemed correct, it must be incorporated into case memory as a new case, thus allowing the reasoner to learn. In section 1.2.2, case indexing was mentioned as one of the issues pertaining to case memory. In this section, what is being referred to is the issue of ensuring that, on being presented with a complete case to incorporate (the solved input problem), the case is added correctly to case memory. Ie, it is incorporated in such a way that when the knowledge it contains is relevant to a new input problem, it will be found in case memory, just as cases were retrieved when it was being solved.

This means that at storage time, the important features of the new case must be identified, and the case must be labelled and stored according to these relevant features. In a flat memory system with global weighting (eg the TFM system (Bayles & Das, 1994); the experiments of Petrak, Trappl and Furnkranz (1994)), this would involve no work, as no case-specific indexing is performed. On the other hand, a reasoner involving hierarchies and generalizations, (eg CHEF (Hammond, 1989)), would need to effectively identify and classify the relevant knowledge in the new case, while GRAND (Oosthuizen, 1994), where memory is organized for speed, would have to restructure memory.

The indexing mechanism is discussed further in section 1.4.2, and in detail in Chapter 3, as it is within the scope of this dissertation.

1.3 The Case-Based Reasoning Process

As with components of a CBR system, the literature differs on the number of stages, the exact content of each stage, and the terms used to refer to stages in the CBR process. This variation appears not just in papers which present diagrams of particular implementations, but also in papers which give generalised diagrams of the CBR process. As such, no one approach can be viewed as the only correct one. With this in mind, it was decided to present two generalised diagrams from the literature which give a useful description of the CBR process. The first of these is Figure 1.1, and is taken from Riesbeck & Schank (1989), while the second, Figure 1.2, is taken from Aamodt and Plaza (1994).

As can be seen, the two diagrams bear little superficial resemblance to each other. However, closer examination reveals that the process presented is the same, but the terminology and level of detail is different. Figure 1.1 shows more detail than Figure 1.2, which allows useful linking back to our discussions in section 1.2, while Figure 1.2 attempts to highlight the main processes more clearly. For the purposes of the dissertation, the process will be split into the stages Obtain an Input Problem, Retrieve Appropriate Cases, Construct a Solution, Test the Solution, Repair the Solution, Index and Save the New Case. The following sections contain brief discussions of each.

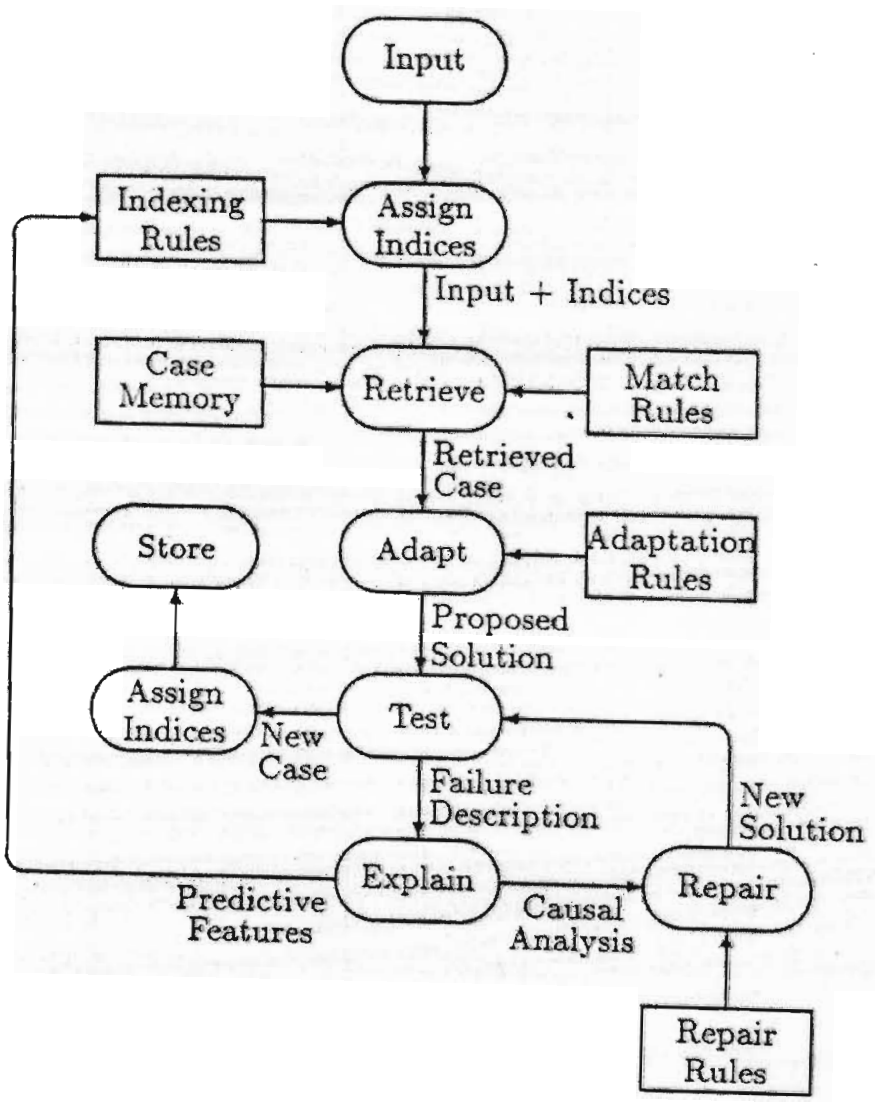

Figure 1.1 : Generalised CBR Cycle taken from Riesbeck and Schank (1989)

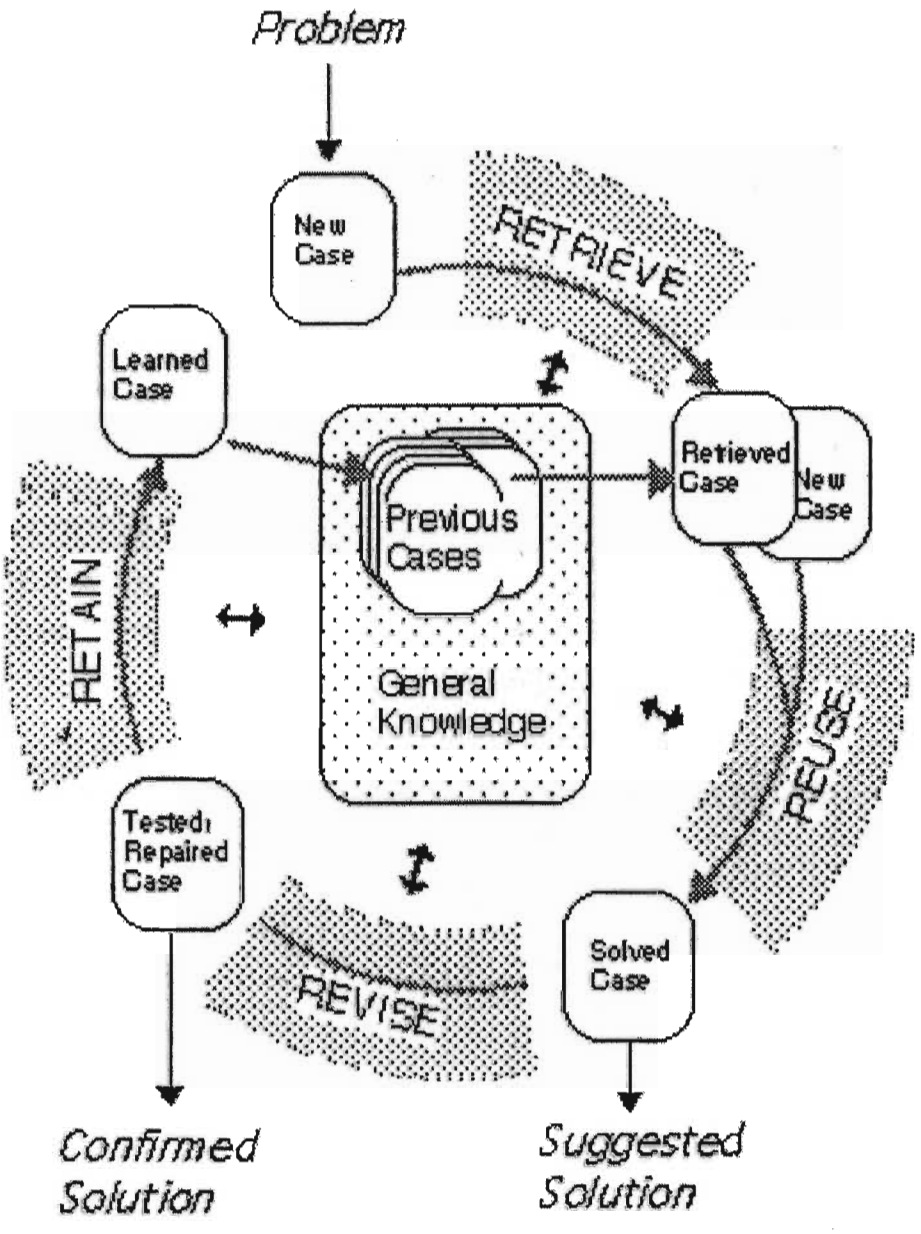

Figure 1.2 : Generalised CBR Cycle taken from Aamodt and Plaza (1994)

1.3.1 Obtain an Input Problem

The input problem is a description of the problem which the CBR system is required to solve. As the process of solving in CBR revolves around finding matching cases in case memory, it follows that the input problem should be in the form of an incomplete case, which can be used to access cases in case memory. More specifically (as mentioned in section 1.2.2), the CBR system should be able to extract the information from an input problem which is needed to access cases in case memory. In the higher level diagram of Figure 1.2, this detail is not specifically highlighted, while in Figure 1.1, this step, the Assign Indices step, is specifically mentioned. In section 1.2.2, it was noted that labelling of cases for storage (indexing) and the extraction of important features in an input problem were both inferencing mechanism issues. In Figure 1.1 this is illustrated by the fact that indexing rules are used to assign indexes to the input problem, thus identifying the important features of the input problem or case.

In systems using a global weighting of cases, (eg the work of Petrak, Trappl and Furnkranz (1994), the TFM system (Bayles & Das, 1994)), the items in an input problem which hold interest are pre-defined due to the global weighting scheme, and as such no extraction of information from the input problem is needed. In a system where cases are indexed in memory based on case-specific information, it might be necessary to extract certain information from the input problem in order to query case memory. Judge (Riesbeck & Schank, 1989) has an analysis phase prior to search of case memory which attempts to interpret the actors motives for actions, and then uses these to search case memory for suitable cases.

1.3.2 Retrieve Appropriate Cases

Once the input problem has been entered, and (possibly) the relevant information extracted, the CBR system will then proceed to search case memory for cases which hold some relevance to the current problem by using the extracted information. In a flat memory system, this would involve fetching each case from case memory for comparison to the input problem. However, memory organisation strategies (for retrieval or speed), as mentioned in section 1.2.1, result in case memory not being a flat structure. In such instances, the indexing mechanism guides the search

to a specific case or set of cases, rather than examining each and every case. CHEF (Hammond, 1989) for example uses generalisations as a means to organise case memory and guide retrieval.

1.3.3 Construct a Solution

As mentioned in section 1.2.4, once the relevant cases have been retrieved, a solution to the current problem must be constructed using the retrieved information. In an automated CBR system, this would involve using a set of stored rules to modify retrieved cases until a match to the input problem was produced, and then modify the retrieved cases solution(s) in a similar manner so as to provide a solution to the input problem (Riesbeck & Schank, 1989). This step is usually referred to in CBR literature as adaptation. Some examples of reasoners which perform automated adaptation are CHEF (Hammond, 1989), which uses generalisations and domain rules to adapt cases, and CLAVIER (Hennessy & Hinkle, 1992), which performs adaptation using pieces of existing cases, rather than domain knowledge encoded in rule form.

In some systems however (eg Battle Planner (Goodman, 1989); the work of Petrak, Trappl & Furnkranz (1994)) the automatic adaptation step, and hence the provision of a new solution, is not performed. Instead, the user is presented with the retrieved cases, but no solution for the input problem. It is then up to the user to produce a solution using the retrieved cases to assist in solution construction. In such situations, the decision making process still relies on the expertise of the user, and the computer system acts as an assistant in the decision making process, helping the user by providing information which they might otherwise have deemed unimportant or else simply overlooked. The reasons for the omitting of this step, according to Kolodner (1991) are issues such as the difficulty of implementing a case adaptor, a lack of need for one. This difficulty of implementation, which in turn could make solutions less reliable, would thus possibly also make adaptation unsuitable for certain mission-critical domains.

1.3.4 Test the Solution

The solution produced by a CBR system is not always optima. By the very nature of CBR, whereby new cases are learned, there is every likelihood that a CBR system might not have

knowledge exactly matching the input problem. As such, there is a chance that the solution constructed by the reasoner (or the human expert based on the retrieved data) will be incorrect, or at least sub-optimal. As a result of this, solutions obtained in the CBR process cannot simply be assimilated into the case base (i.e. learned). It is essential that they are first evaluated to ensure that they are correct, and to identify and rectify those parts of them that are incorrect or non-optimal.

Thus, once the system has produced (or assisted in producing) a solution, this solution must then be evaluated. A system such as CHEF (Hammond, 1989) has a simulated environment on which a solved case (a generated recipe) is tested for functionality. For many reasoners however, this evaluation would have to be performed by the user.

1.3.5 Repair the Solution

If the evaluation process finds that the solution produced is incorrect, or can be improved upon, then a process of modification of the solution is implemented. If the CBR system is one which builds solutions itself, it may have built into it some mechanism for modifying the already produced solution. CHEF (Hammond, 1989) for example uses a set of inference rules which allows it to explain failures in generated plans, and from these explanations generate solutions. However, often it will be necessary for the human expert to assist in the modification process, or else take it over completely.

It is desirable for the CBR system to remember which parts of the produced case failed, why they failed, and how they were modified. This allows the reasoner to avoid repeating errors, or warn against possible failures when it encounters situations similar to one where the plan failed.

1.3.6 Index and Save the New Case

Once the solution has been completed satisfactorily, the reasoner then saves the new solution in case memory. It is at this stage that the indexing mechanism discussed in Section 1.2.4 comes into play. The new case, with its constructed solution and any information on possible errors made,

is labelled (indexed) to identify any identifying information which may be useful to later reasoning. It is then incorporated into case memory based on these indexes.

It is in this way that one of the most important features of CBR, learning, occurs. By incorporating this new case into case memory, the CBR system is effectively learning a new case, thus gaining more knowledge about its domain. This is one of the major advantages of CBR, and will be discussed shortly. Note here that this is learning in the CBR context, where new instance knowledge is learned, rather than generalizations being produced.

The features of the CBR process as discussed in section 1.3 determine the areas of research in the field. It is these areas which are the subject of the next section.

1.4 Research Areas

As with section 1.2 where it was pointed out that the classification of components is not consistent in the literature, so with research areas organisation and naming of research areas tends to vary in the literature. As such, as with components, the classification of research areas used in this dissertation should not be considered the only valid one, nor should it be considered invalid due to variations in the literature. With this in mind, from the literature (eg Barletta (1991), Ketler (1993), Kolodner (1993), Gupta (1994), Aamodt & Plaza (1994) and Leake (1996)) it can generally be identified that there are four major research areas to be addressed when developing a case-based reasoner, namely case representation, indexing, case retrieval, and case adaptation and solution construction.

As the scope of this research requires, case representation, indexing and case retrieval are covered in chapters 2 (case representation) and 3 (indexing, and case retrieval with reference to indexing issues). As such, the discussion of these sections here is in effect a summary of the details discussed in chapters 2 and 3, and detailed discussions are not included. Bearing in mind also that case adaptation and solution construction is not the focus of the research, this section is not covered in great detail either.

1.4.1 Case Representation

Case representation is the most fundamental issue in case-based reasoning. As the method for representing real-world knowledge in the system, all subsequent reasoning techniques must be based on the case representation, and as such their success will depend on case representation. Ascertaining the correct case features was viewed by Barletta (1991) as the most important task in building a CBR system. This serves to highlight the importance of case representation.

Physically, representation of cases varies from database records (Battle Planner (Goodman, 1989; Kolodner, 1993); the TFM system (Bayles & Das, 1994)) to frame representations (eg. ARCHIE (Pearce, Goel, Kolodner, Zimring, Sentosa & Billington, 1992; Kolodner, 1993); CHEF (Hammond, 1989; Kolodner, 1993) to a semantic network approach (eg. Aamodt, 1993). However, it is not this physical structure of cases which is important so much as their content. Case content can be divided into three sections (Kolodner, 1991; 1993), these being problem representation, solution and outcome.

1.4.1.1 Problem Representation

In CBR, the recall of relevant past cases is achieved by comparing the attributes of an input problem to stored cases. As a major area of CBR application is problem solving (which will be discussed shortly), it is unsurprising that cases contain a problem representation, where information describing the problem is stored. This information is used to search case memory for matching cases, and is generally divided into :

Goals - The goals to be achieved in this problem. Eg in ARCHIE, case specific goals such as the importance of group interactions are listed.

Constraints - The specific conditions placed on the problem. In ARCHIE again, this includes requirements such as the height of partitions.

Features - Other problem features which might not be directly necessary for goal achievement. Exactly what this entails is open to interpretation, and is discussed further in section 2.2.1.3.

1.4.1.2 Solution

In CBR systems where a problem is to be solved, it is necessary to present a problem solution. As such, it is necessary to store in each case a solution component which describes the solution for that particular case. Kolodner (1993) split the solution representation into two possible fields :

- The set of justifications for decisions made.
- The reasoning steps used in solving the problem.

The justifications provide the link in the case between the problem representation and solution. These allow for effective re-use of cases, as well in some cases as guiding adaptation, eg CYCLOPS (Navinchandra, 1988). The reasoning steps can be used to assist in deriving a solution to a new problem where the contents of the stored case do not match the input problem. JULIA (Kolodner, 1987; Hinrichs, 1988; Shinn, 1988) records its reasoning steps in its case representation.

1.4.1.3 Case Outcome

The importance of learning in a CBR system has already been highlighted. In addition, it has been discussed how learning is achieved by incorporating new cases. Justifications in the solution section of a case provide the means for the indexing mechanism to store the case in memory, and later identify it correctly. However, a case with only a problem a solution section gives no indication of whether or not a solution was a success. Case outcome holds such information, and provides the reasoner with the capability to avoid future failures.

In CLAVIER (Hennessy & Hinkle, 1993), new stored cases are updated with statistics on the results of their processing in the autoclave. Included in these statistics is the relative success of the layout proposed, and whether or not the layout is good enough for reuse. It is proposed in Hennessy & Hinkle (1993) that this result be used to predict whether or not future proposed layouts would be successful. If such outcomes were not stored, this failure handling would not be possible, as there would be no means of identifying the success of a proposed case solution. As such, if a case was still incorporated into case memory (with an unidentified failure), the solution might be reused at a later stage as though it was correct, thus perpetuating the error.

1.4.2 Indexing

The indexing problem is the issue of ensuring that the correct cases are available at the appropriate times (Kolodner, 1991). While this issue is the basic premise of CBR, there are two areas which are generally given specific attention. The first of these is the assigning of labels to cases to identify them for later reasoning, while the second is the organising of cases in case memory. The organising of cases in case memory is often grouped under retrieval rather than indexing (eg. Barletta, 1991). However, it has implications for both headings. In terms of indexing, case memory organisation is often an integral part. For example, the hierarchical design of CHEF (Hammond, 1989) is essential to the location of correct cases. Thus, memory organisation can be viewed under the heading of indexing.

The classification of indexing methods varies in the literature. However, examination of the literature (eg. Barletta, 1991; Kolodner, 1993; Hansen *et al.*, 1994) reveals that despite different naming conventions, the principles covered are the same. It has been decided here to split indexing techniques into three sections, namely static techniques, inductive techniques and knowledge guided techniques. The following paragraphs give a short overview of these techniques. Details are given in section 3.3 of chapter 3.

Static techniques are fixed techniques (as the name implies) such as nearest neighbour or checklist based indexing. Here, the case memory is indexed using some global indexing method (or methods) without variation of importance of various features being taken into account at a case level (eg Petrak *et al*, 1994). Inductive techniques are implemented by providing a set of representative cases to an inductive algorithm (Hansen *et al.*, 1994). The discriminating features of the cases are determined algorithmically by comparing cases in the representative set, and cases are arranged in case memory based on these discriminating features. Battle Planner (Goodman, 1989) used such a method after a fixed weighting approach proved unsatisfactory. In knowledge-based indexing, domain knowledge is used to identify the predictive features of each case separately (eg CLAVIER (Hennessy & Hinkle, 1992)). All three techniques have their pros and cons, depending on the situation at hand.

Memory organisation strategies, as mentioned earlier, fall under both the indexing and retrieval research areas. In the dissertation, as retrieval strategies were not given particular focus, memory organisation was assessed under the heading of indexing. The literature is once again somewhat unclear as to what all the divisions in memory organisation are. Based on the literature, a mix of ideas from Kolodner (1993) and Barletta (1991) to classify memory organisation, separating the techniques into associative memory, hierarchical memory and discrimination networks, was followed.

With associative memory, case features are indexed independently. Here, memory is generally flat, and is typified by a nearest neighbour approach to indexing (eg CBA (Gonzalez & Laureano-Ortiz, 1992) or Petrak *et al* (1994)). With hierarchical organisation, generalisations of cases are produced, and cases are organised into groups based on shared general features. Discrimination networks appear to produce much the same effect as hierarchical organisation, in other words the grouping of common cases, with generalised case or feature sets being used to access specific cases. Kolodner (1993) stated that the difference is one of purpose, where hierarchies try to cluster cases into small groups, whereas discrimination networks are attempting more to discriminate between cases. Nevertheless, the result is very similar, thus making this distinction difficult to note. Some examples of reasoners which use a hierarchical or network approach are GRAND (Oosthuizen, 1994), PERSUADER (Sycara, 1991; 1993) and CHEF (Hammond, 1989).

1.4.3 Case Retrieval

If one examines the literature for retrieval issues, one will see that techniques for organising memory are examined under the heading of case retrieval (eg. Barletta, 1991; Kolodner, 1993). In addition, we have seen that memory organisation is applicable to the indexing problem (i.e. a focus on the location of the correct cases), and has been discussed (briefly) as such. However, memory organisation is also applicable to the issue of retrieval speed, rather than retrieval accuracy. Battle Planner (Goodman, 1989) for instance used memory organisation both to improve accuracy and speed of retrieval. As such, we see again the overlap of research issues in CBR, i.e. the same topic can be applicable to different issues.

From the literature reviewed it is felt that the discrimination between memory organisation for indexing and memory organisation for retrieval is one of focus, i.e. issues pertaining to memory organisation could be relevant to indexing (case identification) or retrieval (speed of search), and the focus of the research (i.e. speed or case location) decides where these issues are discussed.

To illustrate this point, take CHEF and GRAND (Oosthuizen, 1994) as examples. In CHEF, generalised indexes are used to group cases in memory. In this case, the generalised indexes are being used to allow cases to be applicable in more situations (eg. the BEEF-WITH-GREEN-BEANS being used to create a recipe for the BEEF-AND-BROCCOLI problem). In other words, case memory is being organised to assist indexing. In GRAND, a lattice is used to share features between cases. Here, memory is again being organised. However, this is not being done to assist indexing. Instead, it is being performed to assist in speeding up of retrieval. The point of this example is simply that the focus of memory organisation determines where memory organisation should be placed as a research topic.

As the retrieval of the correct cases as being part of the indexing problem has been covered under indexing, here case retrieval is referring to speed of retrieval. It appears that a hierarchical approach is followed to speed up retrieval (Barletta, 1991), as shown in Oosthuizen (1994). Here, as mentioned, a lattice is used to share features between cases, and greatly speed up search.. Stottler, Henke & King (1989) propose tree structures for organising combinations of attributes in memory.

However, issues of speed and ensuring that correct cases are found (indexing) are unsurprisingly at odds with each other (Barletta, 1991), thus the choice of memory organisation for speed must take into account the indexing problem.

1.4.4 Adaptation and Solution Construction

Ultimately, the goal in any CBR system would be to provide complete solutions to input problems. In an ideal situation, the reasoner would contain a case which was an exact match to

the input problem. In this situation, solution provision is a simple matter of copying the solution from the retrieved case to the input problem.

However, this is often not possible. As discussed, CBR is a learning paradigm. As such, at some stage the reasoner will encounter an input problem for which it does not have an exact match in case memory. In such a situation, the CBR system cannot simply copy a solution from one of the retrieved cases. Instead, if it does attempt to construct a solution, it has to achieve this by adapting these past solutions until they match the new problem.

Barletta (1991) indicated that this adaptation is generally achieved by means of a set of adaptation rules. However, Barletta (1991) also pointed out that in some domains it is difficult to define enough of this rule-like knowledge. In Section 1.6, it will be discussed that CBR is suited to ill-defined domains which are not suited to rule extraction. Leake (1996) pointed out the difficulty associated with acquiring this type of domain knowledge was one of the issues that CBR was aimed at avoiding. What this implies is that the goal of adaptation requires knowledge which one might often not be able to obtain. Kolodner (1991) indicates that the issue of adaptation and solution construction is a highly complex one, and one which can be omitted from reasoners. As such, adaptation in many domains might not be a viable proposal for real-world implementation. Even if knowledge is available, the domain might be of such a nature that attempting case adaptation might be considered highly risky. In such situations, CBR can provide a decision support role, as discussed by Kolodner (1991).

Despite the potential drawbacks of adaptation, reasoners such as CHEF (Hammond, 1989) and CLAVIER (Hennessy & Hinkle, 1992) show that adaptation can be implemented successfully. However, they also illustrate the complexity of the task, and the need for well defined domain knowledge.

As with other research areas of CBR, the terminology used for classifying adaptation techniques is not consistent in the literature. A useful and simple classification is that used in Aamodt &

Plaza (1994), where two adaptation strategies are identified, namely transformational reuse and derivational reuse.

Transformational reuse refers to the concept that the solution in a retrieved case can be modified using pre-defined rules. These pre-defined rules have to be built using extensive domain knowledge which as discussed can be difficult to acquire. The simplest form of transformational reuse would simply be where some of the values in the retrieved solution would be substituted with new values. CHEF (Hammond, 1989) uses this method in initially generating the BEEF-AND-BROCCOLI recipe, as it generates the recipe by substituting the green beans with broccoli in the retrieved case BEEF-WITH-GREEN-BEANS. This however is only part of CHEF's adaptation mechanism, which is far more detailed, and makes use of failures to assist in adaptation and solution generation.

In derivational reuse, the method used for deriving the solution to the retrieved problem is used to produce a solution to the input problem. CoBRA (Cunningham & Slattery, 1993) applies the reasoning trace in a stored case to an input problem in order to produce a solution. The process used is described as derivational analogy, where in replaying the reasoning trace, differences between the retrieved and input cases are taken into account. JULIA (Kolodner, 1987; Hinrichs, 1988; Shinn, 1988) also records its reasoning steps in its case representation. It makes use of the reasoning steps through abstractional analogy, whereby an abstract schema is created from the specific reasoning steps of the retrieved case. This is then applied to the input problem to produce a solution for it.

Other methods of adaptation are also used. CLAVIER (Hennessy & Hinkle, 1992), performs adaptation by pieces of existing cases, rather than using rules to transform retrieved cases. DIAL (Leake, Kinley & Wilson, 1997) uses CBR techniques within a CBR system to enhance an initial set of basic transformation rules. New adaptation rules called adaptation cases, which are traces of memory search, are learnt and reused when adaptation is required again.

As has been discussed, adaptation can be achieved successfully, but it requires domain knowledge which in some cases is not available, as well as having associated risks in certain domains may be unacceptable. As such, it is an important area of research in CBR, but not a essential implementation for all CBR systems.

1.5 Advantages and Disadvantages of Case-Based Reasoning

When discussing the advantages and disadvantages of CBR, the literature (eg Barletta, 1991; Helton, 1991; Slade, 1991; Kolodner & Mark, 1992; Kolodner, 1993; Liang, 1993; Ketler, 1993; Allen, 1994; Gupta, 1994) tends to compare CBR to Rule-Based Reasoning (RBR) more than any other technique. The most likely reason for this is that RBR is the method which has generally been employed when real-world expert systems have been implemented. As such, CBR as a technique with supposed real-world applicability should be compared against the real-world standard, RBR.

Two different areas are discussed when comparing CBR to RBR (or any other technique). The first is in what ways CBR is better/worse than another technique. The second is how CBR offers application in different domains.

One way in which CBR is claimed to hold advantages over RBR is in system building. Many sources (see eg. Barletta, 1991; Ketler, 1993; Allen, 1994) stated that CBR allows for easier and quicker system building than RBR concepts. One of the features highlighted in the literature which makes CBR more favourable than RBR from a system building point of view is in the knowledge extraction phase (Barletta, 1991; Ketler, 1993; Gupta, 1994). In terms of obtaining of knowledge from domain experts, it is reasoned that as CBR provides a cognitive model of the manner in which humans reason, it is easier to obtain knowledge from human experts in the form of cases than rules. From an existing documentation point of view, it is more likely that documentation in an organisation will be in a form resembling cases than in sets of rules. Thus,

little time has to be spent translating existing information into rules, as would be the case if building a rule base.

Barletta (1991) pointed out that time spent on finding the cases cannot be considered extra time, as this work has to be performed in RBR development too. However, Pearce *et al.* (1992) pointed out that complete real-world cases can be difficult to obtain. This indicates that while in theory a CBR system can be simpler to build than a RBR, in reality gathering of knowledge in the form of cases rather than as abstracted rules has its associated problems which must be considered.

A second system building advantage highlighted is the compiling of gathered knowledge into an actual system. Barletta (1991) pointed out that while some shells provide tools to help with rule structuring, a large amount of work still has to be put into rule structuring, whereas in CBR, this is not necessarily the case. The CBR shell REMIND is put forward as an example of time-saving, as it automatically builds a structure for the CBR system. This, however, may be an oversimplification of reality, since while certainly CBR systems do not have the problem of arranging rules, it could quite easily be claimed by RBR proponents that in building a CBR system, this time would be spent deciding on matters such as case storage, indexing, retrieval and learning. Certainly all of these tasks are non-trivial, especially where a system such as REMIND would not be employed. Ketler (1993) pointed out that building an indexing mechanism could well be difficult, but that once it is done, creation of a case base is rapid.

Evidence to back up the idea that CBR allows easier system building than RBR is found in the example of CANASTA and CASCADE given in Kolodner (1993). CANASTA is a RBR system for diagnosis, while CASCADE is the CBR system with the same functionality. CANASTA took 960 person days to complete, while CASCADE took 105 days of development to achieve the same functionality. This type of example illustrates how in certain domains, CBR offers significant advantages over RBR in system building.

Certainly, CBR does offer one distinct advantage over RBR in terms of structuring (and functionality), and this is in the area of long-term system maintenance. This advantage is in the

learning mechanism of CBR, which is documented in much of the literature, (eg. Barletta, 1991; Ketler, 1993; Gupta, 1994). In a RBR system, long term maintenance of the system can be difficult. In order for a RBR system to gain new knowledge, new rules must be added to the system, or existing rules must be modified. This task can be a difficult and time-consuming task, which would have to be performed by the knowledge engineer. In a CBR system, adding new knowledge is a far simpler task. All the knowledge engineering is done at design time. When new cases are added, the already-built indexing and learning mechanism incorporates the new knowledge automatically. This is one of the most important advantages of CBR systems over RBR. Learning (in the CBR sense) offers significant application advantages, as the system can have an incomplete knowledge base, and yet still operational, as well as being able to adapt to changes in a dynamic environment. Another advantage of learning as highlighted in Allen (1994) is that adding of cases is done by the end-user, thus there is no need for a knowledge engineer to add information to the system.

Learning can be looked at as an advantage not only from its widening of the systems applicability, but also from a work-saving perspective (Barletta, 1991). In a RBR system, if the same problem is presented to the system a number of times, exactly the same steps will be followed each time to solve the problem i.e. the system does exactly the same amount of work each time because it is not learning. In contrast to this, when a CBR system has solved a problem once, if it encounters the same problem again, it will simply retrieve the solution, instead of having to construct a solution again. i.e. through learning of more instance knowledge (i.e. CBR learning), the CBR system does less work. However, if one looks at the nature of the systems, this claim must not be accepted without question.

If we look at the operation of a CBR system, we see that once it solves a problem for the first time, in subsequent requests for the same problem to be solved, the stored solution can be recalled, rather than having to re-solve the problem. Thus, in the sense of the operation of the CBR system, its performance improves relative to its earlier performance as less work has to be done. However, we cannot simply say that this means that it involves less work than a RBR system because its performance improves. If we look at a RBR system, we can say (as will be examined later) that

the stored solution (rules) is already optimal, and needs no improving. Thus, provided the RBR solves the problem, there is nothing to be learned. Contrast this to CBR, where the system is still learning the optimal answer. It seems unfair to state that CBR improving its performance for a given solution is better than a RBR, as in reality the RBR already has the optimal solution that the CBR system is still seeking to produce. One might in fact rather say that in a situation where the RBR system produces a solution, it is better than the CBR system until the CBR has learned the optimal solution.

Linked to the idea of ease of maintenance, Barletta (1991) also claimed that a CBR system's inductive powers can be changed by simply resetting the current classifications to new ones, and then re-running the indexing mechanism, while in RBR, the system may have to be re-built from scratch. Once again, this would as a general idea appear to be true, but it might also be looked at as an oversimplification of what may in fact be a decidedly non-trivial problem.

In terms of actual system functionality, CBR offers a further advantage over RBR. The issue that some CBR systems are built to adapt past solutions to solve a new problem has already been addressed. This gives CBR a distinct advantage over RBR in certain domains (Gupta, 1994). A RBR system can only give solutions for problems already coded into its rules. A CBR system which reasons via analogy can to some extent handle problems as yet not encountered. However, this again is not a clear-cut advantage. As mentioned, due to the complexity of the issue, many CBR systems do not implement adaptation. Even successes such as CLAVIER (Hennessy & Hinkle, 1992) found significant problems in implementing adaptation. As such, while adaptation is functionally an advantage, it could be viewed from a building point of view as a disadvantage of CBR.

CBR systems are also said to offer advantages over RBR as far as usage is concerned. It has been mentioned before, and is pointed out in literature such as Kolodner (1991), Slade (1991) and Ketler (1993) that CBR replicates the manner in which human decision making occurs. It then follows from this that human users find a CBR system more intuitive to use than RBR due to the matching of their own reasoning methods. As a result, CBR systems should be easier for end-users

to use and understand (Allen, 1994). In addition, a very simple advantage identified by Barletta (1991) and Kolodner (1991) is that a CBR system helps a human expert remember cases which they otherwise might have forgotten.

CBR systems also show an advantage in that they provide a ready means of explaining and/or justifying the decisions made by the system (Barletta, 1991; Ketler, 1993). In a RBR system, typically, only the chain of rules used to achieve the solution could be presented as justification for the provided solution. CBR systems on the other hand by their very nature provide a means for explaining their actions, this being the cases retrieved for the current input problem.

Another tangible advantage provided by CBR is one discussed already, namely the identification of potential failures. It has already been discussed how the nature of CBR allows for learning of failures which allows avoidance of repeating these failures. Sycara (1993) reported that this feature of CBR also makes it a very attractive AI technique.

All these advantages of CBR over RBR might seem impressive. However, it is good to not lose sight of how a RBR system would work. In a RBR system, all knowledge about the application domain is codified as rules which are theoretically optimal. As such, if a RBR system has had the knowledge built into it to solve a particular problem, the chances are that the solution will be a best (or best at the time of system building) solution i.e. RBR systems are built such that if a solution is produced at all, it will be the optimal one known at building time. CBR, on the other hand, with its tendency towards use of general knowledge and a possibly incomplete knowledge base, as well as the fact that not all possible solutions are examined in solving a problem (Kolodner, 1991), will often tend to provide a good rather than optimal answer. While this performance improves over time, there is still usually no guarantee that the optimal answer is being produced.

Thus, what could be derived from this (in a general sense obviously) is that if a RBR system can be built to handle the application domain, the chances are that the resulting system would produce better solutions than a CBR system. This could lead to the conclusion that, if it is possible to build

a rule-based system for a particular application, it will probably produce better solutions than a case-based system.

This advantage of RBR over CBR highlights the fact that the advantages of RBR over CBR (and in fact CBR over RBR) lie in the domains for which they are applicable. As such, this discussion of CBR will be concluded by briefly examining the domains for which CBR is deemed applicable.

1.6 Domains of Applicability

The literature (eg Kolodner (1991; 1993); Leake, 1996) indicates that there are two major types of reasoners, namely problem solvers and interpretive reasoners. Problem solvers use old solutions as a guide to solve new problems. As such, they operate in much the same overall domain as RBR systems. Interpretive reasoners on the other hand evaluate new situations in the context of old. Clearly, such a technique depends heavily on use of past experience. Here we can immediately see how CBR opens up domains not handled well by RBR.

If we look at the application domain of CBR in a more general sense, and compare this to RBR, it can be seen that CBR is applicable to areas not readily applicable to RBR. The literature indicates that CBR is particularly applicable to domains which are experience rich but knowledge poor (Chi, Chen & Kiang, 1993; Gupta, 1994). In terms of tasks, CBR is said to be suited to those which require creativity and common sense (Kolodner 1991; Gupta, 1994). It is said to be suited to problems where situations dynamically change, where there is much unknown information and solutions are not always clear-cut (Kolodner, 1991), and domains which are poorly understood, and knowledge might even be contradictory (Ketler, 1993).

In order to build well-defined rules, an obvious requirement is that extensive, well-defined and complete knowledge about a domain is available. As such, domains listed above are areas traditionally not satisfactorily handled by RBR. It can thus be said (as the literature claims) that CBR enables the expansion of expert systems applications to areas such as those listed above.

However, here it is important to note that this should not necessarily be looked at as an advantage of CBR over RBR, but rather a difference. The reason for this is as follows.

At the end of the previous section, it was discussed how if a RBR system produces a solution for a problem, the chances are that this solution will be closer to optimal than the solution built by a CBR system which produces satisfactory rather than necessarily optimal solutions. It was discussed that what could be concluded from this is that if a RBR can be built to handle the application domain, the chances are that the resulting system would produce better solutions than a CBR system. What this indicates is that while CBR opens up domains which are not handled well by RBR techniques, the reverse is also to some extent true, namely that where RBR techniques are applicable, they will generally be more optimal than a CBR system in the same domain.

From this discussion, one could, therefore, conclude, as was by Ketler (1993), that CBR will not replace RBR as an expert systems technique, but rather that it is a complementary technique which will allow for expert systems implementations in domains not satisfactorily handled by other AI technologies. CBR, with its ease of building and intuitive use, will allow better or easier applications in certain areas, along with opening up of new application areas. RBR will continue to dominate certain domains, while hybrid techniques, combining rules, cases and other techniques, will develop to solve new problems (Allen, 1994).

1.7 Conclusion to Chapter 1

In this chapter we have seen that case-based reasoning is an artificial intelligence technique which has arisen due to problems associated with other artificial intelligence techniques in combination with a desire to replicate the human reasoning process. Using a case memory of past cases as a knowledge base, case-based reasoning solves problems by retrieving stored cases which contain information relevant to the current problem. As a reasoner based on experience, it holds a number of advantages over traditional artificial intelligence techniques such as rule-based reasoning, not the least of which is automatic learning of knowledge. In addition, it opens up new domains of

application not previously handled well by artificial intelligence, and provides a complementary paradigm to rule-based reasoning.

2. CASE REPRESENTATION

In Artificial Intelligence, the issue of knowledge representation is one of great significance. Without an effective method for representing and storing knowledge, a Knowledge-based System would be unfeasible, as all expertise and reasoning is based on the effective use of this captured knowledge. In CBR, the case is the fundamental method of representing real-world knowledge in a machine-useful format. These cases are central to CBR, as they contain the knowledge needed for solving problems (Gupta, 1994).

Barletta (1991) stated that "Determining the appropriate case features is the main knowledge-engineering task in CBR systems." This statement serves to underline how important the issue of case representation is to the success of a CBR system.

While the above statement conveys the importance of case representation, as we will see in this chapter, this statement possibly falls short of conveying the full extent of the issues involved in designing a case representation format for a CBR system. Issues such as case structure, possible case components and contents of components all come into the picture.

Kolodner (1993) gave the following definition of a case :

> "A case is a contextualized piece of knowledge representing an experience that teaches a lesson fundamental to achieving the goals of the reasoner."

This definition encapsulates the basic concepts which are to be kept in mind in case design, namely that :

- A case stores information about a particular experience.
- That only information which is useful to the reasoner should be stored.
- That the case should be available in situations where it will be useful.

In case design, it is these concepts which determine what representation is chosen. A review of the literature reveals that there are no hard-and-fast rules stating that for a particular application

a particular representation format should be followed. It also reveals that the representation methods vary widely, from database records (eg. Battle Planner (Goodman, 1989; Kolodner, 1993); the TFM system (Bayles & Das, 1994)) to frame representations (eg. ARCHIE (Pearce *et al.*, 1992; Kolodner, 1993); CHEF (Hammond, 1989; Kolodner, 1993)) to a semantic network approach (eg. Aamodt, 1993).

While each choice is applicable to the particular domain, this variability illustrates the point that the important issue in case representation is not the physical structure of the case (eg. frame, database record, etc), but rather the content of the case. When designing a representation, there are a number of issues to address in order to determine the knowledge content of the case. As these issues are more important than the physical case representation used, this chapter's main focus will be on the issues. Following a discussion of them, a brief review of the major representation techniques will be given.

2.1 Principles of Case Building

Kolodner's (1993) definition of a case indicated that a case consists of two elements :

- Information (i.e. the lesson(s) it teaches).
- The context in which the lesson(s) is taught.

The information is the case itself, while the context of a case's lesson is indicated (to the reasoner) by the manner in which the case is indexed. As Chapter 3 covers the indexing problem, this chapter will focus only on the information to be stored in a case, leaving the discussion on context to the next chapter on indexing.

The preceding paragraph already highlights one of the major difficulties of separating the study of CBR techniques into categories (i.e. case representation, indexing, etc). This is that, as discussed in Chapter 1, the different areas in a CBR system tend to be highly integrated, with the reasoning for the design of one section in a reasoner directly supporting, and being supported by, the other areas i.e., we cannot really look at the case representation of any specific reasoner totally in isolation as this will have been designed in conjunction with the indexing method(s) employed.

The literature is indicative of this situation. It is seldom that a paper addresses one specific area, without at least mentioning other areas. For example, a paper concentrating on indexing or retrieval will invariably at least touch on how case representation was undertaken in the system, (as is seen in Selfridge & Cuthill (1989) or Wall, Donahue & Hill (1988)), as the representation of cases (as well as indexing) will have a direct influence on how (and how successfully) search and retrieval is implemented.

Thus, while case representation will be discussed here, and indexing in the next chapter, it is important to remember that these areas are closely linked and cannot be regarded as completely separate entities.

As mentioned at the start of the chapter, cases are the means of storing knowledge in a case-based system, and are the basis for success. Also discussed was that all elements of a CBR system are closely connected, and that cases are designed towards the purpose of the reasoner, i.e. is the reasoner a problem solver or an interpreter?; will cases be used for adaptation?; etc. All of these issues lead to very application-specific case representation. Thus, as discussed, it is the satisfactory addressing of these issues which is more important than the actual representation.

If one looks at the possible structure of a case, one can find three possible case components (Kolodner, 1991; 1993). For the purposes of this work, the issues of knowledge representation will be addressed through a discussion of these case components in the following section.

2.2 Components of a Case

Kolodner (1993) and Kolodner (1991) provided comprehensive overviews of the components of a case. The principles presented in this section are thus to a large extent gleaned from Kolodner's findings, with illustrations of various points taken from the literature.

In the previous chapter the fact that CBR is based on the idea that human experts reason from past experience was presented. It thus seems obvious that a case should be an encoding of a past

experience, i.e. a description of a problem, and its solution. However, for a theoretically complete CBR system, this representation is inadequate, for the following reason.

One of the major advantages of a CBR is the ability to learn new solutions. However, if in problem solving a reasoner only stores problem descriptions and solutions, a problem can occur with respect to incorrect solutions. If such a reasoner produces a solution which is incorrect, and then learns this solution without noting the fact that it is incorrect, it will then propose this incorrect solution as a solution in later problem solving, i.e. the reasoner will provide incorrect solutions. This error has occurred because the reasoner has no knowledge of the outcome of its proposed solutions. Thus, unless there is external control ensuring that only correct solutions are remembered, the knowledge base of a reasoner will quickly become corrupt.

The method used to avoid the re-use of past failures as solutions is to store the outcome of a solution once it has been tested in the real world. This allows the reasoner to avoid repeating the same mistake, as well as learn more about it's operating domain. This concept of failure-driven learning is discussed often in the literature (Slade, 1991; Gupta, 1994).

From the above discussion, the following three major case components can be identified, as discussed and listed in Kolodner (1991; 1993) :

- Problem Description.
- Solution.
- Outcome.

In the following sections, these three components will be discussed in more detail. It should be kept in mind, however, and will be shown during the discussion, that not all components, or parts of components, are needed in each reasoner. Instead, the application domain determines which components are included. This will be discussed in more detail after the discussion of the three components.

2.2.1 The Problem Representation

This is the first section in any case representation. The remembering task in CBR is achieved by comparing a new situation to past situations using some matching method. It is the problem representation component which is used in remembering a past case, and in determining its applicability to a new situation. The problem representation is often not used in isolation. For example, Battle Planner retrieves cases on a problem description as well as a solution description. Nevertheless, the problem description is usually the basis for retrieval.

As noted earlier, indexing is used to present the context of a case. Thus, while noting that the problem description is used for retrieval, the specifics of case retrieval will not be discussed here. Instead, the discussion will be restricted to the actual knowledge needed in the problem description in order that it may be effectively identified.

Kolodner (1993) identified three elements needed in a problem representation :

- Goals to be achieved.
- Goal constraints.
- Features of the situation.

2.2.1.1 Goals

These are obviously the aims to be achieved in this particular case. These could be "abstract or concrete, overarching or narrow" (Kolodner 1993). For example, a concrete, overarching goal might be to create a recipe (as in CHEF (Hammond, 1989)), propose a layout for parts in an autoclave (CLAVIER (Hennessy & Hinkle, 1992)) or planning a battle (Battle Planner (Goodman, 1989)). A more abstract overarching goal might be to understand, such as AQUA (Ram, 1989) which attempts to understand newspaper stories on terrorism in the Middle East, or labour mediation such as in PERSUADER (Sycara, 1991; Sycara, 1993). More narrow goals would be sub-goals such as remembering and adapting, or else the specific goals of a particular problem.

A good illustration of these narrower goals is found in ARCHIE (Pearce *et al*, 1992). Here, the overarching goal is to assist in architectural design. However, the goal in each design is different, thus case goals are included in the case description. This goes to the level in ARCHIE that the entire problem description section is composed of the goals of a particular design.

The recording and type of goals in a problem description are determined largely by the purpose of the reasoner, and will be discussed shortly.

2.2.1.2 Constraints

Constraints are the conditions placed on the goals of the reasoner. A good illustration of a constraint is that in CHEF (Hammond, 1989), where the goal is to create a recipe. In each situation, the requirement is more complex than this - not just any recipe is required, but instead, a specific one. The specifications of the recipe (eg. ingredients) are the constraints on the goal. Thus, beef and broccoli are constraints on said recipe. It appears that constraints can be viewed as lowest level goals.

2.2.1.3 Features of the Situation

This is all other features of the problem description which are not covered in the other two elements. Kolodner (1993) indicated that these are generally perceived on case entry to not be absolutely necessary for goal achievement, but rather that they might hold some information needed for achieving the goals.

This may be open to interpretation though, depending on one's interpretation of goals and constraints. As an example, let us look at CASCADE (Simoudis, 1992). Here, CBR is used to suggest solutions when a customer's VMS operating system crashes, i.e. the overarching goal is to provide a solution. Constraints on this goal (depending on one's interpretation of the term) would possibly be information such as the equipment on which the system was running, as this may constrain the development of a solution. The features, however, would be the actual description of the problem. While these are not goals, they are most certainly essential to the achieving of the goals, as without them, matching cases cannot be retrieved.

What can be concluded from this is that the exact definition of different sections of the representation can become open to some interpretation. However, the combination of these three elements is all the facts needed to describe a case in the context of a particular reasoner's needs. In fact, it is not necessary that a particular reasoner incorporate all these elements as only some elements are necessary in achieving certain tasks. The main point is that the problem representation must be sufficient to allow efficient comparison between cases, as well as distinguishing cases from each other in terms of the specific application.

For example, the problem representation section of ARCHIE only stores goals, as these goals completely define a particular situation for the purposes of the reasoner. On the other hand, Battle Planner concentrates its problem description on features of a situation, as it needs these details in order to successfully evaluate a battle plan. CLAVIER, having a well defined and overarching goal, takes in constraints, namely priority with which each part in the autoclave is needed, as well as features, namely the parts to be arranged, in its problem representation.

A last point which can be made regarding problem representation is that the information included should be useful information. For example, if a system's overarching goal is to produce a recipe (eg. CHEF), there is no point in storing this goal in each case. If it is implied that a case's goal is to produce a recipe, it follows that storing "goal: create recipe" would not only be a waste of storage space but also of processing time, as in each matching performed in the system the goal matching will always be true.

Another example of not storing unnecessary knowledge in cases is found in COOKIE (McCartney, 1993). Here, information regarding the execution environment of recipes (i.e. the kitchen and kitchen implements) is not specifically encoded in each case. Instead this is stored separately and made available to all cases, thus saving on storage space for cases.

The above discussion has given specific examples of the point that unnecessary information should not be stored in a case. In general, information which does not differentiate one case from another, or is not used in solution building, should ideally not be included. However, the point

goes further than simply that the information should be able to distinguish the case from others. The information should also be useful to the reasoning task. For example, if a reasoner storing previous medical diagnoses stored the weather on the day of the diagnosis as a feature in the problem description, this feature would differentiate between cases. However, this differentiation is not meaningful, as (presumably) weather has no influence on a diagnosis. This example is obviously an extreme, but nevertheless illustrates the point.

Kolodner (1993) stated that problem solvers emphasize goals and constraints while interpreters concentrate more on features. This, however, is admitted to be a generalisation, and a review of CBR systems will show that, as with all parts of a system, problem representation varies widely. Kolodner (1993) suggested two general guidelines :

- Include all descriptive information explicitly taken into account in achieving goals.
- Include all descriptive information normally used to describe this sort of case.

The first guideline is obvious as this is the information most important when two cases are compared. The second is used when the first fails, as well as in weak-theory domains, where we actually are not sure of the links between descriptions and solutions (i.e. we don't know whether or not weather affects a diagnosis!).

Generally, it should be kept in mind that the problem description is used to identify a particular problem, and should thus contain all information which could be used to effectively identify the case when it is relevant, and to differentiate it from other cases in case memory.

2.2.2 Solution Content

In order for a CBR system to solve problems, it is necessary that a solution is included in a case description. This is stored in the solution component. As with problem representation, the solution description can consist of a number of elements. The first of these is a description of the solution itself. In the ideal situation for a CBR system, an exact match would be found between a new problem and a stored case. In such a situation, the solution from the stored case could simply be presented as the solution for the new case. However, it is often the case that an exact match is not

found for a new problem. It is in these situations that more knowledge needs to be encoded in the solution component.

The type of knowledge needed by the reasoner can be viewed as the reasoner's understanding of a case. If the basis of CBR, namely human reasoning, is examined, one can see where understanding plays a role. In order to utilise old knowledge, a human reasoner must not only know the facts of a situation, but also understand them. Without this understanding, the knowledge is useless to the expert.

This concept of understanding is implemented in a case in two areas : Solutions and Outcomes. In both areas, understanding in terms of the CBR system is achieved by encoding explanations for actions and results in the two components.

Specifically, in the solution component, understanding is achieved by incorporating an explanation for a solution's derivation in the solution representation. Kolodner (1993) split this into two sections :

- The set of justifications for decisions made.
- The reasoning steps used in solving the problem.

Justification for decisions made is the more obvious method which can be used for allowing reuse of old situations. If an explanation for why each part of that solution was produced is included in a solution description, it can become a relatively simple task to use that solution (or a part of it) in a new situation i.e. if in a solution it is indicated that a certain combination of factors in the problem description led to a certain decision being made, then if in the new problem the same set of factors is encountered, it can be deduced that the same decision may be made here. This use of justifications can be used not only for direct transfer of solution, but also for adaptation, as is the case in the Demand Posting technique in CYCLOPS (Navinchandra, 1988).

CYCLOPS needs to be able to identify solutions in cases for particular sub-goals within the case i.e. it needs an indication within a case for why a particular solution/part solution was used. This

use of explanations allows parts of solutions from stored cases to be applied to a new problem. For example, the placing of a house on a slope is achieved by using stilts, which is learned from a previous case where stilts were used to raise houses off the ground to avoid floods. The sub-goal of keeping the house off the ground was used to locate the stored case. In this way, the handling of a problem even when it has not before been encountered in its entirety is achieved.

The inclusion of reasoning steps can be helpful when an old case's solution is not applicable to the new problem. Here, instead of using the old solution, it might still be possible to take the method used to derive a solution for the old case, and use it to produce a solution for the new problem, thus making an old case useful despite the fact that it does not appear directly applicable to the new problem. An example of this is JULIA (Kolodner, 1987; Hinrichs, 1988; Shinn, 1988), which records its reasoning steps in its case representation.

When JULIA fails to find a case which can provide a directly transferable solution, it makes use of the reasoning steps used in the nearest matching case to produce a solution to the input problem. This is done by the process of abstractional analogy, whereby an abstract schema is created from the specific reasoning steps of the retrieved case. This is then applied to the input problem to produce a solution for it.

Apart from the explanation for a solution, Kolodner (1993) also recommended the following two possible elements of solutions :

- Possible alternate solutions, both those acceptable but rejected, and unacceptable. While these might not be easily used in a fully automated reasoner, they might well be useful in a decision support reasoner which provides cases rather than solutions.
- Expectations about outcomes, including possibly expected problems. These would be useful in later re-interpreting the difference between a new problem and a retrieved case by comparing the actual outcome to the expected outcome as based on the retrieved case's expectations.

In this discussion, a number of issues regarding the content of a solution description have been considered. The most important matter to remember when designing the solution content is that the solution should contain the knowledge needed to perform its necessary task in the reasoner which, for most problem-solving reasoners, means the provision of information to allow the construction of a solution for a new problem.

2.2.3 Case Outcome

In the ideal situation, the solution/evaluation produced by a CBR system is completely correct. In such a situation, there is no need for the reasoner to learn the outcome or reasons for the outcome, as the knowledge needed to repeat the success is already contained within the reasoner. However, if the reasoner fails, it is important that it receives feedback about its failure. The reasoner must make use of this feedback to correct its misunderstanding of the application domain which led to the recommendation of the incorrect solution, thus enabling it to avoid a repetition of the same error.

Thus, as discussed earlier, in order to learn effectively, a reasoner needs to store case outcomes. In the previous section, the issue of understanding was discussed, and it was stated that the means of a reasoner understanding a case is through the use of explanations. The case outcome section is used to encode information about how the solution changed the world. If the change was as expected, outcome encoding is fairly trivial as there are no unexpected outcomes to explain. If, however, a violation of expectations occurred, there is a need for the reasoner to understand this violation. It is in these situations that the outcome component becomes most useful.

A point to remember is that a violation of expectations can be viewed as a failure (even if it led to a real-world success) from a CBR point of view, as it is the result of an incorrect solution provision by the reasoner. Thus, it could be said that the explanation associated with an outcome is an explanation of failure.

A possible full outcome listing for a reasoner could consist of the following elements as listed in Kolodner (1993) :

- The outcome itself.
- Whether it violated expectations.
- Whether it failed or succeeded.
- An explanation of the failure/violation.
- A repair strategy.
- How the problem could have been avoided.
- A pointer to a case where the repair was applied.

The outcome description itself obviously depends largely on the domain and goals. For instance, it may be as simple as "Succeed" or "Fail". Alternatively, it could be a more complex description of the world after solution implementation, or the occurrences as a result of the solution implementation. A good example of this is CHEF (Hammond, 1989), where the reasoner's outcome expectations for the plan to produce a stir-fried beef and broccoli dish include issues such as taste (salty, savoury), texture (broccoli is crisp), etc.

In this case, one of these outcome expectations is violated, namely that the broccoli is not crisp, which results in a change of name to BAD-BEEF-AND-BROCCOLI. This represents an error in the reasoner's understanding of the domain. If expectations are violated, such as happens in this example, this indicates that a reasoner is missing information, as its predictions are inaccurate. Giving the precise faults can allow the reasoner to learn the information to avoid repeating the mistake. This is where explanation comes in. It is not sufficient to simply indicate failures and/or violations. It is only if explanations for these violations are included that the CBR system can learn to avoid the same mistakes in future. This is achieved in CHEF through construction of an explanation for the cause of the violation.

Thus, if we store a case which failed, along with the reasons for its failure, when a new problem matches this case, we now can use the failure learning to indicate how previously a similar situation was incorrectly addressed and, thus, avoid another failure. In addition, a recording of how the failure can be avoided will allow for a successful solution to the new problem.

A repair strategy alternatively can be used to recover from other failures, or to suggest alternative solutions. CHEF constructs a repair strategy to recover from the soggy broccoli problem. This repair strategy is then available for later re-use in failure recovery. This plan is obviously a back-up plan, as it is preferable for the reasoner to identify a potential failure and avoid it rather than recover from one.

Lastly, a pointer to follow-up attempts could be used to indicate where successful solutions lie, although, hopefully, correct indexing and search techniques would allow the finding of these follow-up attempts anyway.

Thus, in outcome representation, the underlying goal to be remembered is that the case must be able to indicate whether the solution succeeded or failed, thus ensuring that only correct solutions are labelled as correct and proposed later, and that new potential failures can be warned against by remembering past failures.

2.3 How Components Affect Use of a Case

Having discussed the various components of cases, it is useful to examine shortly when these components should be included in a case representation. Earlier, the linking of all aspects of a reasoner was alluded to, along with the extreme diversity in case representation. Thus, the use of these components obviously varies according to the application domain for a reasoner. This variation can be looked at from two levels. The first is the split between problem solvers and interpreters. The second level regards the activities which a reasoner covers. These are issues such as case retrieval, learning, evaluation, failure avoidance, repair of failures, adaptation, and derivation of new solutions.

CASEY (Kolodner, 1993) is a good example of a problem solving reasoner which does not store its outcomes. CASEY concentrates on producing a solution to a new problem, but does no evaluation or failure avoidance. As such, it stores only the first two case components - problem

descriptions and solutions. It shows us that for problem solving, a case must store a problem description and a solution.

As discussed in section 2.2.3, this can lead to problems, as there is no noting of success or failure. Thus, if the goal is to produce a reasoner which can learn and refine its knowledge base, it is necessary to include an outcome section, such as is found in a reasoner like ARCHIE (Pearce *et al*, 1992). This gives the reasoner the capability to learn about its environment, and avoid failures as discussed earlier. ARCHIE does not actually construct solutions, so here outcomes are not used for improving the system's own ability to produce solutions. Nevertheless, the outcomes are used to critique constructed solutions. It is thus shown here that in order for effective solution evaluation, outcome storage is necessary.

CHEF (Hammond, 1989), being a more independent problem solver, gives a more complete representation of cases and, as discussed, makes use of outcomes to evaluate plans, avoid or recover from failures. The explanations given for solutions and outcomes enable the last task in CBR, namely adaptation of cases.

The second major area of CBR systems, namely interpretive reasoners, makes a different use of components. For instance, Battle Planner makes use of a problem description and solution to evaluate a battle plan, and indicate possible outcomes. In other words, it does not propose a solution, but instead offers an interpretation for the outcome of a solution. It can thus be said from a CBR perspective that Battle Planner in fact does not use solutions at all, but rather that the 'solution' is part of the problem description. This leads to the concept proposed in Kolodner (1993) that interpreters make use of problem descriptions and outcomes without necessarily needing a solution. Some other reasoners which exhibit this nature are AQUA (Ram, 1989) and OCCAM (Pazzani, 1989).

It can, thus, in conclusion be generalised that for problem solvers, an efficient encoding of at least the problem description and solution is necessary, while if failure avoidance/recovery is an issue, outcome should be included. On the other hand, interpreters need a representation of the problem

description and outcome, but not necessarily the solution. An encoding of the solution may exist, but this is, from the reasoner's perspective, part of the problem description.

2.4 Techniques for Case Representation

Having presented an overview of the principles used for deciding on case representation, a short listing of the major techniques used to represent cases will be given. The literature tends to reveal very little in the way of describing different techniques. In overview-type papers, there is seldom more than a line or two devoted to the topic. In papers of a more specific nature, the tendency is to describe the exact representation used, without necessarily paying much attention to classifying it as any particular type (eg. frame). In addition, representations are extremely application specific. All this indicates that the choice of technique is important only in so far as it achieves the requirements as outlined in the principles. In fact, an attempt to classify representations as being of a specific type may, to some extent, be artificial. As such, this section simply attempts to introduce the major techniques which have been used to represent cases in CBR systems.

2.4.1 Database Representations

In such a representation, a case is encoded in a form-like structure much like a database record. A representation is flat, i.e. has no embedded structures in it. This is an easy method to represent cases, and works in domains where content of cases is well known (Kolodner, 1993). In addition, it is suitable for real-world problem solvers, as it is easy for the user to understand and use. This point is illustrated by the fact that both REMIND and ESTEEM (Watson, 1995), two toolkits for developing CBR systems, are designed to allow this form of representation.

2.4.2 Frame Representations

Frames appear to be the manner in which most CBR systems store their cases. However, this term is to some extent misleading, and is used rather loosely in the literature. For example, Goodman (1989) stated that Battle Planner's cases were built in a frame-based CBR Shell (REMIND), while Kolodner (1993) viewed Battle Planner's cases as being form- or database-like.

One could say that cases are stored as frames in so far as information is stored in slots and sub-slots. However, they do not make use of the functionality of a frame as defined in Turban (1991), where a frame has procedures attached to it. The reason for this is that in a case-based system, case structure tends to be fairly consistent, thus any particular case does not need to store information about the type of knowledge in it, or store procedures, as this can be done on a higher level, namely, the level of the reasoner as a whole. Thus, in CBR it seems that, in general, when a case representation is claimed to use frames, the term is being used loosely to indicate that knowledge in a case is stored in slots and sub-slots, rather than in a flat, database-type structure. We include in this CBR definition of a frame the case representation used in a reasoner such as COOKIE (McCartney, 1993), where the actual account of a meal preparation is included in the case.

2.4.3 Networks

Network representations are used on some reasoners. Such reasoners tend to use a network to represent all the knowledge in case memory, rather than using discrete networks for each case. For example, Aamodt (1993) describes an architecture whereby cases, which could be represented by a frame structure, are incorporated into a semantic network structure. Similarly, GRAND (Oosthuizen, 1994), uses a network to store all cases in a single memory, while a single case viewed in isolation would be seen as a frame. What this points out is that the representation of a single case in isolation is often not how it is actually stored in memory, and as such, often more than one technique for representation of cases is used in a particular system, as is the case in the examples given, where both frame and network technology is used.

2.5 Conclusion to Chapter 2

In this chapter, an overview of the principles governing the implementation of case representation, the requirements of case representation, the components of cases, and the major types of case representation has been given. Without an effective case representation, it would be impossible to implement indexing techniques. As such, it was necessary to investigate these principles and techniques in order that a case representation be implemented which would allow successful

implementation and testing of indexing techniques. In the following chapter, the principles and techniques surrounding indexing, the focus of this research, is performed.

3. INDEXING

The basis of a CBR system's capability to solve problems is the ability to retrieve past situations appropriate to a new situation (Barletta, 1991). Human experts draw analogies between past situations and a present one during problem solving. We have seen that CBR is an attempt to make a computer emulate this retrieval ability (Slade, 1991). The issue of ensuring that the correct cases are retrieved at the correct times is referred to as the indexing problem (Kolodner, 1991).

The indexing problem can be split into two areas according to Kolodner (1993). The first is the assigning of labels to cases to ensure that they are retrieved at the appropriate times. The second is the organising of the case library in such a manner that retrieval of cases is done most efficiently (this will be referred to as memory organization). As we have already discussed, memory organization also falls under the considerations of retrieval algorithms.

These two indexing issues generally address different goals. The first issue addresses the need to ensure that any one case is available for all areas in which it might teach a lesson. In other words, it tries to ensure that the widest range of possibilities is covered. This attempt at high coverage can possibly lead to a speed penalty. The second issue on the other hand attempts to arrange the case library (or rather the indexes) in such a way that retrieval speed is optimized. As with most software design, this improvement in speed can lead to a loss in quality, especially as flexibility is lost (Barletta, 1991). It should be noted, however, that organising the case library (eg. into a hierarchy) is not necessarily only due to speed considerations. CHEF (Hammond, 1989), for example, used generalisations to enable case adaptation. This is an example of memory organisation specifically to assist in indexing rather than to enhance speed of retrieval.

Thus, what can be seen from these two issues is that apart from the most basic issue of the indexing problem, i.e. simply ensuring that cases are labelled in such a manner that they may be retrieved at the correct time, the knowledge engineer is then faced with a second issue, namely, a trade-off between speed and precision. From the literature reviewed, it appears that there is no

one fixed answer to the problem. Rather, the balance struck between speed and precision is (as with most CBR issues) a highly domain-dependent issue.

In Chapter 1, the point was made that while in CBR, research can be split into different areas (eg. case representation, indexing, retrieval, adaptation), these areas are invariably closely interlinked when any particular system is examined. In this essential element of CBR, the retrieval of relevant past cases, this interlinking of research areas is most obvious.

In the context of the two major issues in indexing, the following areas will be covered in this chapter :

The desirable qualities in indexes	-	i.e. the concepts which should be addressed and the necessary information which should be captured when the indexing method for a reasoner is decided on, and when indexes are assigned for a specific case.
Choosing an indexing vocabulary	-	Once it has been decided what knowledge needs to be captured by indexes, it must be decided what vocabulary should actually be used for indexing. This discussion will not cover any particular vocabulary in detail, as this is application specific. Instead, it will concentrate on the requirements of the vocabulary.
Methods for index assigning	-	This is the issue of what technique(s) should be used when actually assigning indexes to a case. This is where the first issue mentioned above will be discussed in detail.
Memory organisation strategies	-	Having covered the concepts of how to label a specific case, the second major issue, namely the major techniques for organising case memory, will be discussed. As this is an overlap between

indexing and retrieval, this area will not be discussed in great detail.

It will be seen that in these two main areas, there are a wide number of issues to consider when designing indexing and memory organization, and that the solutions are largely domain specific. For instance, indexing could be a simple matter of assigning weights to fields in a flat memory structure, as can be seen in Petrak, Trappl and Furnkranz (1994), through to a hierarchical memory organization such as the second implementation of Battle Planner (Goodman, 1989), through to multiple indexes for cases depending on the current goal, as in OCCAM (Pazzani, 1989).

3.1 Desirable Qualities of Indexes

When assessing the desirable qualities for a case's indexes, it should be remembered what the job of indexing a case is - namely to allow it to be identified in a situation where it can teach a useful lesson. Indexing, as with case representation, tends to be domain specific, and the detail and amount of indexing will vary according to needs. Nevertheless, a number of desirable qualities of indexes are identified in Kolodner (1991; 1993). These are predictiveness, abstractness, concreteness and usefulness. These will be discussed briefly in the following sections.

3.1.1 Predictiveness of Features

In Chapter 2 (Case Representation), it was mentioned that a case should not only contain a solution and outcome description, but also an explanation for the content of each. The concept of predictiveness refers to these explanations. Specifically, the parts of a case which were the reason for a solution being derived are predictive of the solution (and are documented as such in the explanation), while aspects responsible for the outcome are predictive of the outcome (again shown by explanation). For example in CHEF, case ingredients could be considered predictive of the solution (the recipe), while if the outcome was a failure, those combinations of ingredients (or perhaps their nature) resulting in this failure would be predictive of the outcome (eg. the failed BEEF-AND-BROCCOLI recipe, a much mentioned case from Hammond (1989)).

The predictive features are obviously the important ones for a case, as they indicate the important elements of the case. Thus, when a case is indexed, it would be desirable for it to be indexed on its predictive features. This may seem to be a relatively easy task when viewed in this simplistic manner. However, it becomes increasingly complex when a reasoner has more than one task, or when it is intended that the system expand, and areas of expansion cannot easily be predicted. As such, there are a number of different techniques for indexing a case, which cover the issue of predictiveness of indexes in a different way, some more completely than others. This is discussed in Section 3.3 on indexing methods.

3.1.2 Abstractness of Indexes

It is often required in reasoners that a case teach lessons in areas other than its specific situation. A good example of this is recipe/meal planning (CHEF and JULIA (Hinrichs, 1988; Shinn, 1988; Kolodner, 1987)). Here, a recipe could be labelled in its specific situation, eg. containing beef and broccoli (as in CHEF). However, it would be very useful to also label it as a meat and crisp vegetable dish, thus allowing the recipe to be used again in a wider area than the specifics of the case. In other words, it would be useful to label the case in an abstract manner. CHEF achieved case adaptation by doing exactly this.

The point of this abstraction is to make a case applicable to the greatest possible range of situations. It is the concept of abstraction which one would consider when attempting to develop a general problem-solving type of reasoner.

3.1.3 Concreteness of Indexes

While abstraction has the advantage of allowing a past solution to be used in a wide number of different areas, thus giving the reasoner wide coverage, there is always the danger that abstraction can be taken too far. In such a case, quick access to cases might be lost, as extensive inferencing would have to be performed simply to access stored cases, no matter how close the real matching. Thus, it is important to ensure that a middle ground is reached, where indexing is abstract enough

to make cases generally applicable, and yet still concrete enough to ensure that cases are readily identifiable.

3.1.4 Usefulness of Indexes

Usefulness of indexes is possibly the most intuitive concept of the four. In any description of a real-world situation (i.e. a case) there are obviously many possible lessons which it could teach. However, unless a specific lesson might be used in later reasoning, it is pointless that it be indexed for that lesson. An example might be as follows: A recipe might fail due to too much of a certain ingredient being added, which occurred because too many people were helping in preparation, which led to miscommunication. This case teaches intuitively two lessons :

- The well known "too many cooks spoil the broth".
- That too much of ingredient X leads to a failed dish.

However, if we have a reasoner for recipe generation, we are not interested in the first point, as we are not worried about staffing problems or proverbs. Thus, indexes must only identify lessons a case teaches which are relevant to the operating domain of the reasoner.

In Chapter 2, the components necessary for the various possible reasoning tasks were looked at. In the same way, we can now say which case components should be indexed on in order to achieve different reasoning tasks :

- For problem solving, a case should be indexed on those descriptors which were responsible for a solution or solution component.
- For evaluation/interpretation, parts of the case responsible for an outcome should be indexed, along with parts which identify the outcome.
- Components responsible for failures should be indexed on to :
  - Anticipate problems.
  - Explain errors and failures in reasoning.
  - Recover from failures (Kolodner, 1993).

A last point of indexing is that while a case should only be indexed for lessons useful to the task at hand, one should not forget that this may well mean that a case may have more than one index

associated with it, as it may teach more than one relevant lesson (eg. OCCAM (Pazzani, 1989), where cases are indexed using two types of indexes, predictive and explanatory).

Having looked at the points to keep in mind when designing indexes for cases, the next issue in the indexing problem is that of index assignment. The first issue here is the choosing of an indexing vocabulary, while the second is that of actual methods for assigning indexes.

3.2 Choosing an Indexing Vocabulary

Due to the fact that an indexing vocabulary is application specific, the literature tends to concentrate on specific vocabularies used for a particular reasoner. As such, there is little published on the general requirements of an indexing vocabulary, other than that in Kolodner (1993). Essentially, Kolodner (1993) identified three guidelines for determining coverage for indexes :

- The vocabulary should be general enough to cover the full range of tasks for the reasoner, yet specific enough to sufficiently differentiate cases. This has, essentially, already been described in the previous section.
- The indexing vocabulary should cover the range of cases to be used
  i.e. a case should not have a value which needs to be indexed which cannot be handled by the indexing vocabulary.
- The indexing vocabulary should be designed in anticipation of future expansion of the system.

In other words, indexes should be able to index all present and future cases in a system in such a manner as to achieve all the goals of the system as effectively as possible.

In addition, Kolodner(1993) suggested two possible methods for selecting an indexing vocabulary for a reasoner, namely the functional and reminding approaches. In the functional approach, an available set of cases is examined in order to determine what the vocabulary should be, while in the reminding approach, the remindings of human experts are examined, and important descriptors

which need to be indexed on are decided in this manner. The choice of method is obviously domain specific.

3.3 Indexing Methods

Views on actual methods for assigning indexes to cases tend to vary widely in the literature. Kolodner (1993) for example split indexing methods into two main areas, namely indexing by hand, and indexing by machine, which is then further split into indexing using checklists, difference based indexing, explanation based indexing, or a combination of the techniques. Barletta (1991) on the other hand took a different view of methods for index selection. Here, methods are differentiated more according to the available knowledge and type of reasoning goal, and are split into Nearest Neighbour, Inductive and Knowledge Guided approaches. Hansen *et al.* (1994) preferred to split indexing into Surface feature indexing and Structural indexing.

These apparently different ways of classifying, essentially, the same issue are indicative of the great diversity of views in CBR, and in some ways the lack of any formal standards and definitions in the field of CBR. However, examination of the classifications shows that there is a high degree of overlap between the various classifications, which are in some cases referring to the same concepts. For the purposes of this section, the Barletta (1991) method of classifying indexing techniques will be followed, but in a modified form. Kolodner's (1993) and Hansen *et al*'s (1994) classifications will be integrated into the classification presented here.

3.3.1 Fixed Indexing Techniques

3.3.1.1 Nearest Neighbour Approaches (Fixed Matching)

Nearest Neighbour approaches in this context refer to indexing used when there is traversal of a flat case memory (eg. a single data base table), with the problem case being matched to each case in case memory. This method was described in Hansen *et al.* (1994). Cases retrieved are those which most closely match the current problem. Here, indexing is performed deciding on the generally important features of all cases as a group, and then assigning weights to these fields. Thus, when those fields with higher weights match, the overall matching has a higher value, thus

ensuring the selection of the best matches. Note here that cases are not weighted according to individual features, but only at a global level. According to Barletta (1991), this method works in areas where "the retrieval goal is not well defined or if few cases are available." This makes sense, as by examining every case in the case library, as little information as possible is lost.

However, there is a major problem with this approach, as discussed in Wettschereck, Aha & Mohri (1997). This is that in many domains, the importance of various features varies from case to case. Thus, each case should, ideally, have its own set of weights which identify its discriminating features. Exactly such a problem was encountered in Battle Planner (Goodman, 1989). Here, domain experts found it extremely difficult to assign weights to fields when a Nearest Neighbour algorithm was used. As weights were modified to suit one set of cases, so they failed to work effectively on another. The technique was in fact abandoned due to the failure to find a successful consistent weighting scheme. This problem was solved by using an inductive approach which will be discussed shortly.

The shortcomings of weighting the same fields in all cases in case memory are obvious, and there are a number of solutions to the problem, as will be discussed next. It is, however, important to bear in mind that it is still a highly useful technique which has been implemented successfully, eg. Simoudis & Miller (1990); Petrak *et al.* (1994); the CBA system of Gonzalez & Laureano-Ortiz (1992). As stated in Kolodner (1993), shallow indexing of flat memory is by far the most common means of indexing case-based systems. It, thus, should never be discounted as a simple and effective indexing mechanism. There are various means of indexing and matching of cases using a nearest neighbour approach. Petrak *et al.* (1994) gave a good and detailed description of some of these.

3.3.1.2 Using Checklists

Checklist-based indexing (Kolodner, 1993) bears little resemblance to a fixed-weight approach to indexing such as the nearest neighbour search approach. However, there is one feature shared between the two methods, which will be discussed shortly.

In the checklist approach to indexing, at build time a list of features is created on which every case is indexed. Thus, when a new case is indexed, the checklist is referred to, and those fields specified in the checklist are the fields on which the case is indexed. In the case of CHEF (Hammond, 1989), this would mean that all cases are indexed on ingredients, method of preparation, as well as other fields. The point is that every case is indexed on these same fields, namely those identified in the checklist.

It is this fact that in all cases the same fields are indexed that provides common ground with the fixed-weighting nearest neighbour approach. In fixed-weighting, all cases have the same global weighting thus, effectively, all cases are indexed on the same features. This leads to problems of determining good weightings. In a checklist approach, the same type of technique is being used in so far as it is being presumed that the same fields in all cases are teaching the lessons required. What this means is that the domain has to be very thoroughly analysed in order to produce a good checklist, and the same problem as before, namely that cases are not indexed individually, is encountered.

However, checklist indexing is not nearly as restrictive as fixed-weighting nearest neighbour indexing. In the nearest neighbour, the memory structure is flat, and cases are indexed for one goal only. Checklists do not restrict in this manner. For example, there is no reason why a case should only be teaching a single lesson (and thus being indexed once). Multiple goals (therefore multiple indexing) can be achieved simply by creating multiple checklists where each checklist is designed around a different context. In this way, any case can be retrieved for a number of different goals.

In addition, from a memory structure point of view, checklist indexing does not prevent the structuring of memory in a hierarchical manner. A good example of this is CHEF (Hammond, 1989), where indexes are generalised (eg. beef becomes meat), thus memory becomes hierarchical in nature.

Checklist indexing thus has in common with nearest neighbour fixed weighting the concept of fixed feature matching. However, unlike nearest neighbour, the technique is not restrictive in

terms of memory structure, nor does it restrict the number of contexts in which a case may be used to teach a lesson.

3.3.2 Inductive Approaches

Inductive indexing (Structural indexing in Hansen *et al.*, 1994) is based on the use of an algorithm to index cases and arrange memory based on differences between cases. ARCHIE (Pearce *et al.*, 1992) uses nearest neighbour and inductive approaches. Battle Planner's (Goodman, 1989) problems of assigning weights to fields were solved by using inductive indexing. In this approach, indexing is generally performed by providing a representative set of cases to an inductive algorithm (Hansen *et al.*, 1994). This algorithm then analyses the cases and, based on the decision required from the cases, determines which features best discriminate between cases, and creates a decision tree based on these features.

This technique provides a number of advantages, as mentioned in Barletta (1991), and seen in the development of Battle Planner. The first is that cases are automatically analysed for their predictive features. The difficulty of identifying the important features by hand, as was initially done in Battle Planner, are avoided. The second is that memory can be organised hierarchically. This greatly cuts down on the retrieval time of cases, which was initially a problem with Battle Planner too.

It is intuitive that inductive approaches provide advantages over the more simple nearest neighbour weighting approaches as there is a movement from a static indexing method to one generated from the specific requirements of the system. However, this advancement does bring its disadvantages.

Firstly, the goal or outcome must be a well-defined one (Barletta, 1991). Without the definition, it would be impossible to decide which features are best for distinguishing cases according to that definition. Secondly, there must be enough cases to give adequate coverage of each type of goal otherwise comparisons will not be performed effectively (Barletta, 1991). A third drawback identified by Barletta (1991) is that the inductive analysis can take a lot of time. Nevertheless, as

illustrated by Battle Planner, inductive analysis can indeed be used highly successfully to index cases.

Kolodner's (1993) Difference-Based indexing appears to refer to the same basic concept as inductive indexing, namely that indexing is performed through comparing cases. The basis of this theory is that the purpose of indexing is to differentiate between cases (in a manner which supports the reasoning goal(s)). It, thus, makes sense that a possible technique for indexing is to index on the differences which a case exhibits, eg. if a case's value for a particular field is the normal value, do not index on it. However, if this value differs from the norm, it differentiates the case from the norm, and thus should be indexed on this feature.

3.3.3 Knowledge-based Indexing

Knowledge-based Indexing (Explanation-Based Indexing in Kolodner (1993), not covered in Hansen *et al.* (1994)) is another way of overcoming the problems of a fixed weighting nearest neighbour indexing approach, and in fact also inductive/difference-based techniques. In all techniques discussed thus far, there is no analysis of individual cases in order to discover how each is predictive. Even though inductive indexing techniques decide which features of a case are predictive based on initial individual comparison, they do so by making use of a model of features that are usually predictive (Kolodner, 1993), and do not analyse each specific case to determine which features are predictive for that specific case. This means that :

- Features which are generally predictive will be indexed on even in cases where they are not.
- Features which are generally not predictive will not be indexed on even in cases where they are predictive (Kolodner, 1993).

This leads to the situation where inappropriate cases are retrieved, or where appropriate cases are overlooked. This does not necessarily lead to failure, but rather to sub-optimal performance.

The concept of knowledge-based or explanation-based indexing is that each case is analysed individually to determine what features of that case are predictive, and the case is then indexed

on these features. This is achieved by building sufficient explanatory information into the reasoner (Barletta, 1991).

Earlier in Section 3.1 the desirable qualities of indexes were mentioned. It is in the areas of predictiveness and usefulness where knowledge-based indexing offers the greatest improvement over other indexing techniques, and makes it the preferred approach to indexing when the knowledge necessary to implement the technique is available (Barletta 1991), such as in the case of CLAVIER (Hennessy & Hinkle, 1992) or CHEF (Hammond, 1989). If each case is analysed individually for its predictive features, then (provided the explanatory information is correct and complete) a case will be indexed on all of its own specific predictive features, thus achieving the goal of predictiveness, while not be indexed on non-predictive features, thus ensuring that all indexes are useful.

It will have been noticed that in Section 3.3.1, an alternative name was suggested for the Nearest Neighbour indexing approach, namely the Fixed Weighting approach. This is because Barletta's (1991) description of a nearest neighbour approach indicated that each case is indexed via fixed weighting, thus each case is identically indexed. It is contended, however, that a nearest neighbour matching system should not have to be approached using a fixed weight scheme as described by Barletta (1991). Instead, a knowledge guided method could be employed whereby each case is assigned its own weighting. This method was in fact suggested by Barletta (1991) as a desired approach to the nearest neighbour method. However, the link is not explicitly made to knowledge guided indexing methods. It is, thus, contended here that assigning individual case weightings could be achieved in a nearest neighbour system by making use of domain knowledge (if available). Thus, it is possibly misleading to refer to the first method as nearest neighbour approaches, therefore fixed weighting approaches is suggested as an alternative.

There are of course drawbacks to knowledge guided techniques. The first is that the explanatory knowledge needed must be available and representable (Barletta, 1991). The second is that it is often difficult to encode enough of this knowledge to effectively index all possible cases (Barletta, 1991).

However, a solution to the problem was presented in Kolodner's (1993) first indexing technique - indexing by hand, without it being explicitly presented as such. This technique is in fact a knowledge guided technique, simply it is not an automated one. Instead of the reasoner having explanatory knowledge built into it which will be used to analyse a case, it is left to the user to identify for the reasoner the indexes for a case.

The user, therefore, is required to follow the guidelines for desirable qualities of reasoners, and thus must identify the predictive features of the input case (for all possible reasoning tasks), generalise where possible (only if the reasoner is using generalisation, which will be covered later), and then translate these features into the reasoner's indexing vocabulary. While this process might appear tedious, it is felt that it is highly applicable to reasoners which supply decision support as proposed in Kolodner (1991).

If we look at this simplified process, we can see that it is in fact a knowledge (explanation) based approach, with explanations being provided by the user, not the reasoner. This of course overcomes the issues of representing and encoding enough explanatory knowledge.

However, at the same time it brings with it a number of cons. The first is of course this user involvement. The whole essence of the "expert system" is that the computer performs all of the expert processes which of course includes the indexing of new cases. Involving the user removes from the "ideal" expert system. The second is that by involving the user in the learning process, we are opening up the issue to problems of human error. This might be a lack of expertise, carelessness, or even the intentional recommendation of incorrect answers. All of these could lead to incorrect indexing of cases and, thus, corruption of the knowledge base.

3.4 Memory Organization Strategies

Having looked at the principles behind the indexing of specific cases, the last section to look at with regards to the indexing problem is that of memory organization. It is difficult to categorize whether memory organization should be discussed in conjunction with indexing or retrieval

techniques. Kolodner (1993) for example grouped organisation with retrieval. However, as discussed, it is contended that it really falls between the two issues. On the one hand, it provides a specific organisation of cases in memory, thus linking it to indexing as it is being used to differentiate cases. In addition, generalising of cases, as done in CHEF (Hammond, 1989), allows for case adaptation which might otherwise not be successfully achieved. On the other hand, it is linked to retrieval as it is the organising of case memory in a manner which makes accessing of cases most simple and fast, thus linking it to retrieval. As retrieval techniques and algorithms are not within the scope of this research, memory organization will be discussed here due to its intimate link with indexing.

Case memories tend to be organised in three particular ways (Barletta, 1991; Kolodner, 1993) :

- Associative Memory.
- Hierarchical Memory.
- Discrimination Networks.

These will each be discussed in turn.

It will have been noticed that, apart from the section on abstractness of indexes, little until this time has been mentioned of the well-known indexing technique of generalisation. This is because for the purposes of this chapter, the concept of generalisation will fall into this section on memory organisation strategies.

3.4.1 Associative Memory

The definition of an associative memory as given by Barletta (1991) is that "any or all of the features of a case are indexed independently of the other features." This structure is typified by a flat memory, nearest neighbour approach to indexing which for the purposes of this research is being referred to as fixed weighting. In fixed weighting, weights are assigned to fields at build time, and every new case is referenced using those weights. Thus, apart from initial weight determination when the system is built, fields are always indexed independently.

A flat memory organisation such as this is fairly simple, and has various associated advantages and disadvantages.

As mentioned earlier, the technique is good to use in domains where the goal is undefined. Each case is always examined, thus minimising the possibility of missing novel cases. The technique is also simple to implement, and learning of a new case is trivial.

However, there are some serious disadvantages. The major one (Kolodner 1993) is that it is an expensive scheme. Matching against all cases can be effective for a small case base, but ineffective in a large one, due to the large time taken to access all cases. A second consideration is that use of independent indexing means that it is hard to customise the reasoner for several goals. Barletta (1991) appeared to state the opposite, saying that an associative approach should be used for flexible systems. However, this would only appear to be the case in a system where, as mentioned before, goals are undefined. In a system where there are multiple (but known and defined) goals, such a technique might be inappropriate, but not in all cases, as shall be seen in the techniques designed for this dissertation.

Kolodner (1993) suggested an improvement to a flat memory approach, this being a single level of indexing. Here, certain descriptors are chosen as indexes, and these indexes point to the cases containing their descriptor. In matching a new case, those descriptors which are found in the new case are identified, and their cases are retrieved for case matching. In this way, search time can be significantly improved. Such a technique was used in CASCADE (Simoudis, 1992). However, one then stands to lose some flexibility as all cases are no longer being accessed.

Associative, flat memories are thus found to have their advantages and, as mentioned earlier, have been implemented with great success. Some successful examples are CBA (Gonzalez & Laureano-Ortiz, 1992), Simoudis & Miller (1990) and Petrak *et al.* (1994). However, when case libraries are large, and/or when goals are better defined, other techniques (hierarchical and discrimination nets) are recommended.

3.4.2 Hierarchical Organization

This technique makes use of generalisation of indexes by organizing cases in a hierarchical memory structure. As mentioned earlier, inductive approaches can be used to organise cases into a hierarchy. Names such as decision trees and shared feature networks are used to describe the actual memory structures, but they all involve some mechanism for arranging memory in a general-to-specific manner.

For example, in a meal planner, we might classify a meal as having an ingredient of beef. At a higher level this might be a red meat, and even higher simply a meat. The point is that the higher level indexes are used to differentiate the case memory into small chunks. Tree traversal leads to more specific indexes until exact cases are found. The improvement in search time is obvious, as instead of accessing each case, search is rapid tree traversal to a small set of cases, rather than the whole case memory.

Of course, while splitting memory has advantages, it also has its drawbacks. For instance, how do we ensure that the correct cases are retrieved? Theoretically, if the tree is structured correctly, only the necessary cases will be accessed, but there is always the chance that some important lessons may be missed. This leads to a second problem, namely that discriminators may have to be changed i.e. as the system learns, it may be found that the original descriptors are non-optimal (the question of optimality is a separate issue), and the tree may have to be re-structured. A system may in fact require several trees to be constructed in order to satisfy the various goals.

The last disadvantage is the speed of adding a new case. In a flat memory, a case may simply be added to the list. In a hierarchy, however, it must be placed in the correct leaf, thus involving tree traversal. Such a situation is shown in GRAND (Oosthuizen, 1994), where case insertion takes up to twenty minutes, but retrieval time is of the order of a second.

3.4.3 Discrimination Networks

Having discussed hierarchical organization of memory, an important point must be made. According to Kolodner (1993), hierarchical organisation clusters memory with the goal of speedy retrieval. On the other hand, discrimination networks achieve clustering as a by-product of the goal to differentiate cases. As such, it is possible that we should not view hierarchical memory organisation as defined here as being part of indexing. Rather, for indexing purposes, discrimination networks are the hierarchical system to be assessed.

Other than this conceptual or goal difference, there appears to be little to differentiate between the two techniques physically from Kolodner's (1993) perspective. However, what we could say is that the inductive techniques described earlier give rise to discrimination networks. This is based on the idea that inductive algorithms organise memory based on differences between cases, i.e. the goal is to differentiate cases rather than to organise memory in a hierarchy to speed up retrieval.

As such, decision trees and clustering and other hierarchies such as those described in Oosthuizen (1994) or Stottler *et al.* (1989) could be viewed as hierarchies according to the definition used here, as their intention is the speeding up of retrieval. However, the tree produced by induction in Battle Planner would be viewed as a discrimination network due to it being derived in order to improve indexing, even though it happens to improve retrieval time too.

Whatever definition is followed, the generalising of cases in memory to create discrimination networks appears to be a highly popular area of research of CBR. Battle Planner uses inductive discrimination to arrange case memory. ARCHIE (Pearce, Goel, Kolodner, Zimring, Sentosa & Billington, 1992; Kolodner, 1993); makes use of clustering of cases. LAWCLERK (Selfridge & Cuthill, 1989) uses a number of memory structures in memory organisation. CHEF (Hammond, 1989) uses a discrimination network approach, with generalisations allowing adaptation. PERSUADER (Sycara, 1991; Sycara, 1993) uses generalized episodes to organize memory into a discrimination network. JULIA (Hinrichs, 1988; Shinn, 1988; Kolodner, 1987) uses generalized information along with actual cases in the meal-planning domain.

This high degree of interest in discrimination networks is unsurprising for two possible reasons. Firstly, the faults associated with a flat, associative memory structure (as mentioned in section 3.4.1) mean that some other form of memory organisation is needed. It should be remembered, however, that this does not mean that associative memory is wrong, simply that its faults make it inapplicable to certain domains. Secondly, apart from the associative approach, all other memory organisation can be characterised as some type of a network. As such, until more clear standards and definitions appear in the CBR world, these techniques are all grouped under the discrimination network heading.

3.5 Conclusion to Chapter 3

In this chapter, the principles governing indexing, requirements of indexes, various indexing methods, and methods for organizing memory have been covered. As indexing is the focus of the research, it is essential that techniques designed and implemented be based on valid theories. Having reviewed case representation (chapter 2) and indexing, the next chapter provides a discussion of the domain used for applying indexing techniques.

4. REVIEW OF PRACTICAL AND RESEARCH ISSUES RELATED TO PRE-TRANSPORTATION OF HAZARDOUS WASTE

While the focus of this research was on indexing in CBR (and case representation as a necessity), it was felt that the theory should be tested on a practical example. This was to show that CBR, and specifically the techniques designed and tested in this research, was applicable to the building of a pre-transportation decision support tool for hazardous waste handling. As such, two areas were covered. Firstly, this chapter discusses the problem of hazardous waste, and discusses how and why it is a domain in which CBR techniques are applicable and useful. Secondly, all designing (Chapter 5) and testing (Chapter 6) of indexing techniques is done using data from the proposed application domain, and is designed with the application domain in mind.

4.1 The Hazardous Waste Problem

Many aspects of environmental management are currently a matter of global concern. The various issues associated with hazardous waste, from production to handling to transport, final disposal and storage, are in need of particular attention world-wide and in South Africa, as indicated by the Department of Environment Affairs (1992).

According to the same report, of all the environmental problem areas, hazardous waste management has received the least attention in South Africa, and "has been the slowest to develop either direction or regulatory mechanisms". The report highlights the need for an effective regulatory system for all aspects of the handling of hazardous waste. Obviously a report of this nature is referring to the problem at a national level and, as such, problems highlighted and

solutions proposed in this report are operating at a higher level than the scope of this study, namely a specific application for a specific company. Nevertheless, for management to be successful nationally, it is obvious that successful management should also be achieved at the level of individual companies. The Department of Environment Affairs (1992) backed up this idea by indicating that very few industries have waste management strategies. In addition, it is felt that demonstrating the effectiveness of this particular application could lead to further uses for the same technology.

A number of areas were highlighted where hazardous waste should be dealt with. These were :

- Reduce the waste at its source.
- Recycle waste as much as feasible.
- Treat the remaining waste effectively.
- Dispose of the residue effectively.

In addition, it was noted that few professionals in industry or local government are adequately informed about the merits of different solutions available. One of the main themes in the report is that existing practices are often unsatisfactory, and that there is a lack of adequate provision of information. It is also highlighted that there is a lack of detailed information in many organisations on the waste being handled.

The significance of this report is that it indicates the importance of the hazardous waste problem, and justifies research in the field of waste technology. In addition, this study involves a knowledge-intensive application. As the inadequacy of knowledge is one of the problem areas, it is felt that research into a knowledge-intensive technique such as the one discussed here is justified.

4.2 The Applicability of Computing to Hazardous Waste Handling

Internationally, much research attention has been paid to issues of hazardous waste handling. Papers such as Andersson (1994), Haastrup (1994), List & Turnquist (1993) and Beroggi (1994)

serve to show this attention to various aspects of the hazardous waste problem. In particular, much work can be found on the application of various modelling techniques to areas of hazardous waste management. Modelling has been applied to areas such as urban solid waste management (Caruso, Colorni & Paruccini, 1993) and transportation of hazardous waste (List, Mirchandani, Turnquist & Zografos, 1991) as well as other environmental issues.

These papers make two points of relevance to this dissertation. Firstly, hazardous waste management is a subject in need of researching, and one which is viable for research. Secondly, information technology application (so far modelling only has been mentioned) is needed in the area. In addition, the papers show that in the hazardous waste management domain, there is a great deal of information which needs managing. It has already been discussed how CBR is applicable to domains in which there is a large amount of knowledge management needed, especially where the domain is not always well-structured. In a more general sense, it can be said that AI techniques are good at handling large amounts of knowledge effectively. As such, a domain such as hazardous waste handling can be viewed as one with potential for AI application.

Papers such as Rauscher & Hacker (1989), Warwick, Mumford & Norton (1993) and Radermacher *et al.* (1994) showed that AI techniques are applicable at the more general level of environmental management. Others like Krovvidy, Wee, Summers & Coleman (1991), Chan & Tontiwachwuthikul (1995) and Repede & Bernardo (1994) showed the applicability of AI to specific environmental problems. More specifically, Hushon (1991) showed the applicability of AI to hazardous chemical management through overviews of the HaSP and TINIA systems. These reports demonstrated that application of AI technology to hazardous waste management is a viable technique which has already been attempted elsewhere in the world. As such, with the need for attention to hazardous waste in South Africa, it was felt that it would be worthwhile investigating the application of information technology to hazardous waste handling. More specifically, it was decided to apply some form of AI technique to a hazardous waste problem to assist in decision support.

Having decided on the basic technology, Waste-tech (Pty) Ltd was chosen as a possible subject for two major reasons. The first was due to Waste-tech's position as a significant handler of hazardous waste in the industrial sector. The second was due to its close links with the International Centre for Waste Technology (Africa). As a leader in the hazardous waste field, it was felt that this would be a good company to find significant existing expertise and data on which to base a decision support tool.

Having chosen the company, it was then left to choose the actual application domain. Discussions with experts at Waste-tech led to the choosing of the pre-transportation decision making phase of hazardous waste handling as the application choice. As will be described in the next section, this is a knowledge-rich area in Waste-tech which lends itself well to structuring and integration of a decision support tool.

4.3 The Current Decision Making Process for Pre-transportation Handling of Hazardous Waste

The current decision making process in place at Waste-tech revolves around a data sheet such as that shown in Figure 1 in the Appendix. It is important to indicate before describing the process that it was totally uncomputerised, and based totally on human expertise.

4.3.1 The Current Process

When Waste-tech receives a batch of waste from a client, a data sheet is used to record the information related to this sheet. Firstly, the basic client information (name, address, date, etc) is recorded on the data sheet. Following this, the "Process generating waste", "Known/expected constituents", "Description of waste" and "Odours" fields are filled in based on data given by the client, and obvious characteristics of the waste.

Once this has been done, a sample of the waste is passed on to the laboratory for analysis. Testing itself requires expertise (other obviously than that required to perform the tests). Specifically, not

all tests are performed for all wastes. Instead, there are certain basic tests which are always performed. Based on the results of these tests and the content of the pre-laboratory test fields, certain other tests are performed. This is continued until all tests required to be performed have been.

Having captured all the data about the batch, the data is handed over to an expert. It is now the job of the expert to provide solutions for the case. These solutions are contained in the "Disposal instructions", "Disposal site", "Hazchem decal no.", "TREM Card", "Hazard warning" and "Protective clothing" fields. The solutions are determined by examining the contents of the laboratory test and pre-laboratory test fields. An important point to note here for later discussion is that in each situation handled, different laboratory test and pre-laboratory test fields are used to determine the solutions. In other words, not all information is used in solution derivation, but rather, the information used for solution derivation is found in different fields for each new situation.

Once solutions have been produced and executed, the completed data sheet is filed along with other data sheets. There is no computerisation of the process, simply the filing of papers.

4.3.2 Problems with the Current Process

The major point to note about the current decision making process in place is that it is totally uncomputerised. While one certainly cannot say that because a process is uncomputerised it can be considered problematic or flawed, examination of the current process does reveal a number of possible problem areas or areas for improvement which computerisation could aid. These are all related to the point that access to old solutions is a tedious process as it involves retrieval of paper sheets from files, and the requirement that the expert remembers the existence of a past problem solved.

Basically, three major possible problem areas can be found :

- Due to the simple filing of data sheets, old solutions are not used actively in the solving of new problems. Instead, the expertise and memory of the expert handling the current

problem is relied upon. As such, past expertise is not easily available for current decision making.

- As with solutions, problems which have occurred in the past in solutions provided are not actively identified again in new situations. Again, the expert is relied upon to identify these problem areas. Thus, failures which have occurred before are not always remembered.
- When a new person takes over the role of producing solutions, validation of the solutions produced is not easily achieved unless another expert is available. This is because the new people being trained in decision making do not have easy access to past expertise.

These three areas highlight a number of problems. If, for instance, an expert has solved a problem before, and then encounters the problem again, there could be a number of scenarios. Firstly, the expert may not remember having handled the problem before, and thus solves the problem again from scratch. Alternatively, the expert may remember having solved the problem before. However, he/she must now remember where this past solution was filed. If they do not then either they remember the solution anyway, and can simply implement it again, or they cannot remember the details, and thus they must solve the problem all over again. Even if they do remember the details of the solution, if the problem is not a common one, they might still want some sort of validation for their solution, a simple cross-check which would easily be provided by finding the past solution.

Another problem is encountered when different problems are handled by different experts. It is conceivable that different experts might solve problems slightly differently. Without access to other experts solutions, an expert cannot easily check that his/her solution is exactly like that of other experts. In addition, without easy access to past solutions, if an expert encounters a new problem, they will have to solve it from scratch, even if another expert may well have already encountered such a problem.

In terms of problems with solutions produced, once again the lack of access to past solutions could cause problems. Unless an expert specifically remembers a past problem, there is every chance

that mistakes made in the past will be repeated, as the experts are not reminded of past failures which occurred.

The problem of a new person taking over the role of decision maker was illustrated during the course of the research. This occurred when the incumbent expert left, resulting in a new person taking over the role. While this person certainly had expertise, a great deal of extra work had to be done due to a lack of experience with the handling of the problems. Provision and validation of solutions took more time due to the new person's lack of experience.

4.4 Potential Application of a CBR Decision Support System to the Current Decision Making Process

4.4.1 The suitability of the domain to CBR

In the previous three chapters, CBR as an AI technique has been outlined in some detail. It has been discussed that CBR is an AI technique which makes use of past experience to solve or assist in the solving of new problems by making available cases which contain information applicable to the current problem.

In addition, examination of the process and problems described in the previous section shows two points. Firstly, most problems/areas for improvement are the result of not efficiently making use of expertise already gathered. Secondly, the specific application domain is rich in past expertise, stored in the form of data sheets. While these data are not computerised, they do make available exactly the type of knowledge for which CBR is ideally suited.

Thus, we have :

- A problem where :
  - The problem is unstructured.
  - We have access to significant past experience.
  - We have a need to access this information effectively.

- An AI technique which, given past expertise in the form of case, makes the correct information available for problem solving.

As such, it was felt that the application domain would be a worthwhile one to test CBR on.

However, this is not the only reason why the domain is suitable for CBR testing. Examination of the problem shows more justifications for the use of CBR. In Chapter 1, a comparison between CBR and Rule Based Reasoning (RBR) was made. Here, a number of advantages of CBR were listed. Interestingly enough, a number of these advantages are issues here.

Firstly, information in the domain is not completely defined. In the application domain, Waste-tech is generally dealing with mixed wastes. As there are, potentially, endless combinations of wastes, any decision support system applied to the domain would have a need to cater for possible new combinations, which are almost inevitably going to be encountered. In a RBR system, this would require the addition of new rules by a knowledge engineer, whereas with CBR, as discussed, this issue is handled simply by being able to learn new problems and their solutions as they occur. This advantage includes the learning of any possible failures.

Secondly, CBR is said to model the human reasoning process and is, thus, easier for a user to make use of. As attempts were being made to examine techniques which could be easily added to the current decision making process, a technique which would not alienate the users was appropriate.

A third issue is justifications for decisions. It has been discussed that by providing access to actual past solutions, CBR allows for easy justification for solutions produced. As has been shown, both new and experienced decision makers would find validation of solutions useful. CBR would provide this validation through access to past data sheets.

The above advantages all led to the conclusion that the application domain would be suitable for testing CBR techniques.

4.4.2 A Proposed Decision Support System based on CBR

It was felt that it was important that any proposed system based on the techniques developed should allow for easy integration into the current decision making process. It was, thus, felt that it would be unwise to design CBR techniques and conduct experiments which if implemented would require extensive modification of the current decision making process.

A proposed decision support system would thus not change that basic decision making process. As is the case now, handling of waste would be centred around a data sheet. The difference would be the that data sheets would be computerised, and made available at decision making time. For this to occur, a database would be created in which all past data sheets would be stored. Past data sheets relevant to the current problem would then be retrieved and made available to the expert when a new problem was being handled. Intelligent retrieval of these data sheets would be achieved through CBR techniques.

The process would therefore be as follows :

- Initial data about a batch of waste is filled in on the data sheet (i.e. up to "Odours"). Note that this data sheet would now be computerised, rather than on paper.
- Laboratory analysis would be performed, and the results of laboratory analysis filled in.

As we can see, the above steps are unchanged from the normal decision making process, other than the fact that the sheet is computerised. The next step involves the proposed system.

- Based on the information entered in the new data sheet up until this point, data sheets stored in the data base (case base) which contained data relevant to the current problem would be retrieved and presented to the expert.
- The expert would then examine the contents of the data sheet and past data sheets, and produce solutions to the new problem.
- Once successful solutions to the new problem had been produced, the completed data sheet would also be saved in the data base, thus making it available for later reasoning too.

This process shows how the proposed system would assist in solving the problems already outlined. By providing the expert with access to the relevant past data sheets (cases), the problems already outlined would essentially be solved, as :

- Decision makers would now have access to relevant past solution, thus :
  - They remember solutions they produced.
  - They see solutions produced by others.
  - They can reuse old solutions, and validate solutions.
- Past failures are presented in the relevant situation to assist in avoiding the same problem.
- New trainees are able to validate their solutions against those of established past experts, as well as learning old solutions.

It must be pointed out here that the system described is a proposed system only. The objective of this research was specifically to investigate indexing (and case representation as a basic requirement) in CBR. As such, the research on indexing uses the proposed application domain as the basis for the design and testing of indexing techniques. However, as developing a system was not the focus of this research, the decision support system itself was not implemented.

4.5 Conclusion to Chapter Four

In this chapter, discussion has been made of the need for attention to all areas of hazardous waste handling. Past research has shown that application of AI to hazardous waste handling is viable. It has been shown that CBR is a technique which is applicable to the building of an intelligent decision support for system for pre-transportation handling of hazardous waste. It has the potential to solve many of the problems associated with the current decision making process. As such, it is a legitimate domain to use for implementing and testing indexing techniques.

5. DESCRIPTION OF IMPLEMENTED CASE REPRESENTATION AND INDEXING TECHNIQUES

As mentioned earlier, the objective of the dissertation was to investigate indexing (and case representation) for CBR, and to design and test indexing techniques using the pre-transportation decision making process for hazardous waste handling at Waste-tech (Pty) Ltd as the domain for the design and testing. In Chapter 4, the proposed operation for the system was discussed. It should be noted that the proposed system would be a decision support one, thus adaptation, which was not a focus of the research, would not need to be researched.

While the main focus of the CBR research was on indexing, the major work presented in this chapter is the development of a number of indexing techniques. However, in order to test indexing of cases, a method for case representation had to be designed and implemented. As indexing is closely linked to the case representation chosen, attention was also placed on development of a good case representation (hence Chapter 2). However, as case representation techniques were not being evaluated, but rather being developed to facilitate comparison of indexing techniques, one case representation was developed. In the next section, the method chosen for representing the cases in Case Memory will be discussed before moving on to a discussion of indexing.

5.1 Case Representation

As the general theoretical and practical issues pertaining to case representation have been discussed in detail in Chapter 2, they will not be re-visited here. Instead, the case structure chosen will be presented, followed by a discussion on the reasons for choosing the case representation.

5.1.1 Description of Case Structure

It was decided, after consultation with Waste-tech, that the data sheet as currently used by them and presented in Chapter 4 would be used in as unmodified a form as possible for case representation. The reasoning for this was twofold. Firstly, the data sheet format already lent itself well to a database format, which as discussed is a valid case format. Secondly, with a view to possible real-world application, it was decided that unnecessary deviations from the existing data layout would be counter-productive. As a result, construction of a case was a matter of creating a field in the case for every field in the data sheet. There were some modifications necessary to this data sheet, but these will be discussed shortly in Section 5.1.2.

The result of applying this principle can be seen in the case representation given in Table 1 in the Appendix. As will be seen in the discussion on the indexing techniques (Section 5.2), two different applications were used to test the techniques. As such, an application-specific (i.e. code listing or database table) case representation is not given in the table. Instead, the representation in Table 1 of the Appendix is an application-independent listing of the case structure. The actual representation of a case in the two systems will be covered later.

As mentioned, construction of a case representation was not quite as simple as creating a field in the case for every field in a data sheet. Examination of the case structure shows that certain fields in the data sheet are not present in the case representation. These fields were omitted as they did not hold information which was used in solution derivation. As such, it was decided that they would cause unnecessary cluttering of the data in a test situation. Specifically, the fields from the data sheet which were omitted are :

- Lab No.
- Date (all occurrences)
- Address
- Tel.
- Fax
- Contact
- Postal Address

- Expected volumes
- Frequency
- Process generating waste
- Classification of waste
- Consultant
- Depot
- Analyst
- Chemist
- Hazardous Waste Manager

In addition, it can further be seen that certain fields were added to the case representation which were present in the data sheet. These are all labelled as NEW in the Data Sheet Field section of Table 1 in the Appendix. Note that all these fields are SOLUTION rather than REPRESENTATION fields, and that they are all a list of reasons. Section 5.1.2.3 contains the reasoning for these omissions and additions.

For exact details on the case structure, please see Table 1 in the Appendix.

5.1.2 Discussion of Case Structure

The case representation and content described were decided on with the following issues in mind :

- The case representation must allow successful implementation of the proposed indexing techniques.
- It must not radically change the current format of the data sheets from Waste-tech which are essentially used as the cases.
- The method chosen must adhere to and satisfy the principles of case representation as discussed in Chapter 2.

Before discussing the above issues, however, a point about the case structure must be made. Examining a flat case structure such as this, it becomes apparent that normalisation of the structure could be performed relatively easily. However, this was not done for two reasons.

Firstly, normalisation offers no advantage to the actual reasoning process, nor does its omission in any way hamper reasoning. As such, it was omitted as an unnecessary complication. Secondly, the one system used to test techniques was ESTEEM, a commercial CBR tool. This package does not handle linked tables in a relational database, which normalisation would create. Thus, in the interests of consistency amongst tests, it was decided to retain a monolithic structure for cases. Having justified the choice of this monolithic case structure, the issues kept in mind when deciding on case representation and content will now be discussed.

5.1.2.1 Successful Implementation of the Indexing Techniques

As the main thrust of the research was the successful indexing of cases, it is obvious that the case structure must allow indexing to be achieved successfully, a principle already discussed in the dissertation. However, one cannot illustrate how the case structure allows successful implementation of indexing techniques without actually providing a full discussion of the indexing techniques. As such, no discussion of this first issue will be given here. Instead, for a complete discussion of indexing techniques implemented, please see Section 5.2.

5.1.2.2 The Retention of the Current Data Sheet Format

As mentioned in the previous section on case structure, apart from certain exceptions already listed, the case representation matched the structure of a data sheet. While research could well reveal that a vastly different data sheet format would facilitate building of a successful reasoner, the intention was to develop techniques which could integrate into the current decision making process. As such, making use of current data in a form as near as possible to the current one was desirable.

5.1.2.3 Adherence to the Principles of Case Representation

There are a number of aspects to discuss in respect to this issue. However, before discussing these, it is worthwhile to consider again quickly the nature of the information present in a data sheet (See Chapter 4 for an overview). There are essentially two types of data in a data sheet - REPRESENTATION and SOLUTION data (see Table 1 in the Appendix). Basically, the

reasoning process involves gathering REPRESENTATION data, and using it to derive the contents of the SOLUTION fields. This SOLUTION data are derived by means of a domain expert examining the contents of the REPRESENTATION fields, and then deciding what the SOLUTIONs should be. Having recapped the basic reasoning process, the reasons for the choice of a particular case structure will now be discussed. The first two issues to consider pertain to something already mentioned about the case structure, namely the omission of certain fields and addition of others.

In Chapter 2, Kolodner's (1993) definition of a case was given. One of the issues raised is that only information which is useful to the reasoner should be included in a case. It is with this principle in mind that the fields listed previously were omitted from the case representation. As just mentioned, the REPRESENTATION fields hold the information needed to derive the solutions, i.e. they hold the useful information. However, the fields omitted from the case representation hold no knowledge applicable to the reasoning process. In other words, while they are useful to Waste-tech, for the purposes of the reasoning research they are not useful. Thus, by the principle that only information useful to the reasoner should be included in a case, as well as in the interests of space and simplicity, these fields have been omitted from the case representation.

In terms of addition of fields, if we look at a data sheet, and bear in mind that there are two sets of fields, REPRESENTATION and SOLUTION, an important point can be made. As we have seen, the REPRESENTATION fields contain the information needed to decide what solutions to implement. However, nowhere in the data sheet is any link made between the REPRESENTATION and SOLUTION fields. In Chapter 4 it was noted that the same fields are not used in each data sheet to derive solutions, i.e. it depends on the value of a field whether or not it is actually used in deriving a solution.

The result of this is that unless one is an expert, reading a data sheet would give no clue as to why a particular solution had been derived, as we would not know which fields were important.

Obviously, a computer processing a single case would in the same manner not be able to make any definite links between REPRESENTATION and SOLUTION fields.

In Chapter 2 one of the listed requirements of the solution content of a case is an explanation for the solution's derivation (in the form of justifications for decisions made, as well possibly as reasoning steps). As just mentioned, this explanation for solution derivation is what is missing from the data sheet, i.e. no link between REPRESENTATION and SOLUTION fields is present. As a result, the NEW fields have been added to the case representation to create this link between REPRESENTATION and SOLUTION fields by providing explanations for the solutions derivations. More specifically, one such NEW field has been added for each of the SOLUTION fields which contains an actual solution (eg. disposalrn is the NEW field for disposal, the SOLUTION field containing the actual disposal solution(s)). For a particular case, disposalrn would contain the list of REPRESENTATION fields and their values which for that specific case were the reasons for the derivation of the solution(s) listed in the disposal field. The same approach would apply to all other fields marked NEW.

An example of a stored case is given in Figure 5.1 (specifically, case 000021), with the NEW fields values entered. (Note that in the interests of simplicity and conciseness, only those REPRESENTATION fields in the stored case which contain non-null values are listed here). Using our disposalrn example, we can deduce from the contents of the disposalrn field that the reason for disposing of the waste using the method ASHBLEND was because one of the constituents (constit) was OIL. Similarly the reasons for other solutions can be simply identified. An exception is made where reason fields are blank (see sitern, warnrn and clothingrn). In these cases, the default solutions had been adopted, i.e. there was nothing in the case content which warranted any exceptional action. As such, no field value can be identified as a reason for the action being taken, thus the reason field is left blank.

	Field Name	Value
REPRESENTATION FIELDS	Sheetno	000021
	Constit	OIL; WATER;
	Wstate	LIQUID
	Odour	WEAK
	ph	7
	Flamm	NEGATIVE
	Hvymetalsp	NEGATIVE
	Fluoridesp	NEGATIVE
	Chloridesp	NEGATIVE
	Sulphatesp	NEGATIVE
SOLUTION FIELDS	Disposal	ASHBLEND
	Disposalrn (NEW)	Constit : OIL;
	Site	UMLAZI
	Sitern (NEW)	
	Hazchem	7017
	Hazchemrn (NEW)	Wstate : LIQUID;
	TREM	6.12
	TREMrn (NEW)	Wstate : LIQUID; Flamm : NEGATIVE;
	Warn	No Smoking/Naked Flame/Handling
	Warnrn (NEW)	
	Clothing	Goggles, Boots, Gloves
	Clothingrn (NEW)	

Figure 5.1 : Stored Case 000021

It can thus be seen that these NEW fields serve as the explanations required to show which REPRESENTATION fields were responsible in a particular case for the derivation of the solutions for that particular case. Thus, these NEW fields satisfy the requirement that the solution section of the case contain explanations (or justifications) for the solution's derivation.

Having justified the specific changes made to the original data sheet, it is also useful to examine the representation in the context of requirements of case representation as discussed in Chapter 2.

Here, it was outlined that a case can be divided into three sections. As such, case representation will be discussed in relation to these three sections.

5.1.2.3.1 The Problem Description Section

For the system, the problem description section of a case is all the REPRESENTATION fields. Three elements of the problem description were listed in Chapter 2 :

- Goals.
- Constraints.
- Features.

Here it was discussed that, in general, problem solver reasoners emphasize goals and constraints, while interpretive reasoners emphasize features. The system under discussion is clearly a problem-solving one and as such not surprisingly, emphasis is not on features.

In terms of goals, the system has one concrete, overarching goal (although one might consider it to have six), namely to find solutions for the six SOLUTION fields. As such, listing a goal in the case structure would be pointless. What characterises the particular situation is constraints on goals. As with CHEF (Hammond, 1989), where the constraints on the overarching goal of recipe creation are the ingredients, so in the present case the values contained in the REPRESENTATION fields of the case are the constraints on the goal of waste disposal.

Most fields which would be considered to be features (eg. date, laboratory number) have been omitted from the case representation as discussed earlier. In a real-world implementation however, it might be useful to in fact include these fields, not so much from a reasoning perspective, but rather, as noted, because Waste-tech would find them useful for matters other than reasoning in the system. One feature type field included is the Company name. While this is not used in the reasoning process, it most certainly is desirable to keep such a field in the case representation.

As discussed in Chapter 2, the fact that a problem representation concentrates almost solely on constraints while ignoring a listing of goals or features is not a cause for concern, as it is not required that all case representations contain all three elements. Rather, the main goal is that the

problem description section should allow efficient comparison between and differentiation of cases. This means that some reasoners will concentrate on constraints, others on features, others on goals and constraints, and so forth. In the particular design presented here, this goal is achieved effectively using primarily constraints for the problem description, as will be seen in the discussion on indexing techniques in Section 5.2.

5.1.2.3.2 The Solution Section

The discussion on solution content in Chapter 2 listed five possible contents of the solution section, namely :

- Solutions.
- Explanations for Solutions (broken up into) :
    - Justifications for Decisions.
    - Reasoning steps used.
- Possible Alternate Solutions.
- Expectations about outcomes.

Examination of the case structure shows that in the SOLUTION fields, the following are included :

- Solutions.
- Justifications for Solutions.

The inclusion of these justifications (the NEW fields) has already been discussed as the means for providing a link between REPRESENTATION and SOLUTION fields. In Section 5.2 it will seen how these justifications enable the successful indexing of cases, as it allows the location of the predictive features of a case.

The remaining three possible contents could all prove interesting to incorporate in the system. However, as mentioned earlier, the desire in the system was to keep matters as little changed as possible from the current situation. While it certainly might be possible to restructure the cases so as to incorporate these sections, this would create complications which would be counter to the goal of allowing the proposed system to be added easily to the current process.

5.1.2.3.3 The Outcome Section

From the representation scheme, it can be seen that there is no outcome section to the case. The primary reason for this is quite simply that there is no such section in the current data sheet. Currently, Waste-tech does not provide this information at all. Thus, while an outcome section would allow the possible implementation of failure prediction, these data simply were not available. This fact did not influence the results of testing the indexing techniques within this research. However, a real-world implementation in this domain would require attention to this issue, with modification of the data sheets to incorporate this information.

5.2 Indexing

As the main focus of the research was the indexing of cases, unlike in case representation where a single technique was used, a number of indexing techniques were designed and tested. All of these are of the nearest neighbour type from a search point of view, i.e. the input problem is matched against all cases in the case base. However, not all use the common fixed weighting method in nearest neighbour search whereby all fields in a case are given a global weighting. The techniques tested represent more of an evolution from one technique to the next than a set of unrelated techniques.

Keeping with the previous section on the case representation designed, a description of the techniques will be given, following which they will be discussed in the context of the problem at hand and the principles of indexing as discussed in Chapter 3. However, more issues need to be dealt with than in the previous section. Specifically, the issues of field matching, case matching and retrieval of cases also need to be discussed in order to obtain a clear picture of how the systems actually work. Thus, the structure of this section will follow the following course :

- Describe the indexing techniques tested.
- Describe Field and Case matching techniques.
- Describe Retrieval of cases.
- Discuss the indexing techniques.

5.2.1 Description of Indexing Techniques Tested

In Section 5.1 it was shown that a case in the system consists of two sections - REPRESENTATION and SOLUTION. In addition it was shown that the information held in the REPRESENTATION section leads to the solutions in the SOLUTION section. As such, as the goal of indexing is to identify the predictive fields in a case, the goal of indexing in the case base is to identify the REPRESENTATION fields which are predictive of the solutions in the SOLUTION fields of the cases.

As mentioned, the techniques represent somewhat of an evolution from one to the next. In all, there were five techniques implemented and tested. These represent a change in indexing approach as represented in Figure 5.2.

Figure 5.2 : Progression of Designed Indexing Techniques

5.2.1.1 Equal Weight Nearest Neighbour (EWNN)

This is the most basic technique for indexing in CBR. For this technique, all fields in the REPRESENTATION section of a case were assigned a weight of one. In other words, no field was given any more importance than another field. This technique is equivalent to the SIM-EVEN technique described by Petrak *et al.* (1994), and shall be referred to as EWNN in the custom implementation and EWNN-EST for the ESTEEM implementation (this will be discussed later).

5.2.1.2 Fixed Weight Nearest Neighbour Using Expert Judgement (FWNN-EXP)

This technique is equivalent to the SIM-F technique of Petrak *et al.*(1994) or the weighting method initially attempted in Battle Planner (Hammond, 1989), and is also a standard and intuitive method for the nearest neighbour technique. For this technique, a person currently involved in the decision making process at Waste-tech was approached. This person was asked to assign a weight

to each field in the REPRESENTATION section of a case based on its importance relative to all other fields in the REPRESENTATION section. The more important the field, the higher the weight. In other words, each field was given a specific weighting according to its perceived importance.

Table 5.1 :Fixed Weight Nearest Neighbour using Expert Judgement (FWNN-EXP) Field Weightings

Field Name	Weight
coname	1
constit	3
wstate	2
odour	2
ph	4
flamm	4
flash	4
arsenicsp	4
cyanidesp	4
hvymetalsp	4
chromesp	4
sulphidesp	4
cyanide	4
arsenic	4
All Other Lab Tests	2

Table 5.1 shows the weights assigned to the fields. Note that not all fields are listed here. All fields not listed in Table 5.1 have a weight of 2 assigned to them. This technique is referred to as FWNN-EXP.

5.2.1.3 Fixed Weight Nearest Neighbour Using the 80-20 Rule (FWNN-80:20)

This technique offers an interesting comparison to the FWNN-EXP technique. Unlike in FWNN-EXP where fields were weighted by an expert, it was decided here to make use of the 80-20 rule for weighting of fields (the 80-20 rule being the concept that 20% of the data contribute to up to 80% of the information (Mitchell, 1993)). Thus, for this technique, weights were assigned to eight

of the forty problem representation fields. These fields were chosen by the author firstly by taking into account advice from the expert, and secondly by examining the data contained in the 90 data sheets used for case base construction. As such, the fields chosen were influenced heavily by the contents of the test case base. Table 5.2 shows the fields which had weights assigned to them, and the weights they were assigned. All other fields were assigned a weight of zero. This technique is referred to as FWNN-80:20 for the system and FWNN-80:20-EST for the ESTEEM control.

Table 5.2 :Fixed Weight Nearest Neighbour using the 80-20 Rule (FWNN-80:20) Field Weightings

Field Name	Weight
constit	3
wstate	2
odour	2
ph	4
flamm	4
flash	4
hvymetalsp	4
sulphidesp	4
All Others	0

5.2.1.4 Variable Weight Nearest Neighbour (VWNN)

All three techniques presented so far make use of a global weighting scheme. In other words, having examined the case base as a whole, weights are assigned to the various fields once. Each case then gets assigned the same weight for a particular field. This global weighting scheme was abandoned for the following techniques, where cases are instead weighted individually.

In section 5.1.2.3 it was described that a number of fields (all marked NEW in the SOLUTION section) were added to the case representation, and that these hold the information which indicates which fields in the REPRESENTATION section were responsible (according to the domain expert) in that particular case for the solution. The contents of these fields are in fact a ready-made

means for indexing a case. As such, Variable Weight Nearest Neighbour (VWNN), instead of using a global weighting of fields in the REPRESENTATION for indexing, uses the values of the NEW (i.e. reason) fields in the SOLUTION section to determine the indexes for a case.

Effectively, what happens for any case is as follows. Firstly, all REPRESENTATION fields get an initial weighting of zero. To then assign weights to fields, the following process is followed. Each time a REPRESENTATION field and its value are listed in a NEW (reason) field, the weight of that REPRESENTATION field for the case is increased by one. Thus, to index the entire case, each NEW (reason) field is scanned to obtain all REPRESENTATION fields listed, along with the number of times they are listed. This information then weights only the predictive REPRESENTATION fields for that case, the weight determined by the number of occurrences of that REPRESENTATION field in the SOLUTION NEW (reason) fields .

As an example illustrating this theory, Figure 5.3 shows how case 000021 would be weighted using the VWNN technique. Note that as with Figure 5.1, only REPRESENTATION fields which contain non-null values are given. Those with null values can all be considered to have a weight of zero. Note also that each constituent (constit) is assigned a separate weight (i.e. OIL has a weight of one, but WATER has a weight of zero).

Using this method, each case is weighted (and thus indexed) individually, and only on the predictive features for that case. In other words, VWNN has moved away from a global weighting scheme.

	Field Name	Value	Weight
REPRESENTATION FIELDS	Sheetno	000021	0
	Constit	OIL; WATER;	1 0
	Wstate	LIQUID	2
	Odour	WEAK	0
	ph	7	0
	Flamm	NEGATIVE	1
	Hvymetalsp	NEGATIVE	0
	Fluoridesp	NEGATIVE	0
	Chloridesp	NEGATIVE	0
	Sulphatesp	NEGATIVE	0
SOLUTION FIELDS	Disposal	ASHBLEND	
	Disposalrn (NEW)	Constit : OIL;	
	Site	UMLAZI	
	Sitern (NEW)		
	Hazchem	7017	
	Hazchemrn (NEW)	Wstate : LIQUID;	
	TREM	6.12	
	TREMrn (NEW)	Wstate : LIQUID; Flamm : NEGATIVE;	
	Warn	No Smoking/Naked Flame/Handling	
	Warnrn (NEW)		
	Clothing	Goggles, Boots, Gloves	
	Clothingrn (NEW)		

Figure 5.3 : Weighting of Case 000021 using Variable Weight Nearest Neighbour (VWNN)

5.2.1.5 Separate Field Variable Weight (SFVW)

The final technique tested was an extension of VWNN. As with VWNN, Separate Field Variable Weight (SFVW) also makes use of the NEW (reason) fields contents in a case. As such, the principle is the same. The difference (or extension) is as follows.

	Field Name	Value	Weight
REPRESENTATION FIELDS	Sheetno	000021	0
	Constit	OIL; WATER;	0 0
	Wstate	LIQUID	1
	Odour	WEAK	0
	ph	7	0
	Flamm	NEGATIVE	1
	Hvymetalsp	NEGATIVE	0
	Fluoridesp	NEGATIVE	0
	Chloridesp	NEGATIVE	0
	Sulphatesp	NEGATIVE	0
SOLUTION FIELDS	Disposal	ASHBLEND	
	Disposalrn (NEW)	Constit : OIL;	
	Site	UMLAZI	
	Sitern (NEW)		
	Hazchem	7017	
	Hazchemrn (NEW)	Wstate : LIQUID;	
	TREM	6.12	
	TREMrn (NEW)	Wstate : LIQUID; Flamm : NEGATIVE;	
	Warn	No Smoking/Naked Flame/Handling	
	Warnrn (NEW)		
	Clothing	Goggles, Boots, Gloves	
	Clothingrn (NEW)		

Figure 5.4 : Weighting of Case 000021 using Separate Field Variable Weight (SFVW) for TREM SOLUTION

In VWNN, a single case-specific index is created for each case by using the NEW (reason) field contents (as opposed to the first three techniques which use a single, global index). SFVW extends this principle by using these NEW (reason) fields to create multiple case-specific indexes. This is achieved fairly simply as follows.

In VWNN, all NEW (reason) fields are scanned once, and their contents used to weight certain fields in the REPRESENTATION section, thus creating a case-specific (but single) index for the case. In SFVW we simply use each NEW (reason) field separately for weighting of the case. In other words, we take the first NEW (reason) field, examine its contents alone, and then weight the case in the same manner as VWNN. We now have the case weighted for the SOLUTION field which the NEW (reason) field contained reasons for. By weighting separately for each NEW (reason) field, we effectively create a case-specific index for each of the SOLUTION fields of the case. Thus, we can say that SFVW creates multiple, case-specific indexes.

As an illustration, Figure 5.4 shows case 000021 weighted using SFVW for the trem SOLUTION. Note that only wstate and flamm have weights assigned, as they are the only values in the tremrn field.

5.2.2 Field and Case Matching

Having presented the indexing techniques used, the issues of how cases are matched, and how retrieval is performed in the design must still be addressed. This section will concentrate specifically on how case matching occurs. It will be left to the next section on retrieval of cases to show how this matching is then used in conjunction with the five indexing techniques to actually retrieve cases.

The computation of similarity (matching) between an input and stored case in the systems is done by calculating matchings between relevant fields, and then computing a matching for the overall stored case. As such, there are two issues to be addressed :

- Field Similarity.
- Case Similarity.

5.2.2.1 Field Similarity

There are only two types of field matchings used for the system. The first applies to the pH field, while the second applies to all other fields used for matching. Note that for all matchings, 'No Test' is an exceptional field value, and is not mentioned until the end of this section.

Valid results for the pH field are integer values in the range 0-14. A matching between two pH values is computed using the Even-Range technique used in Petrak *et al.* (1994). This uses the formula :

$$similarity(pH_{input}, pH_{stored}) = 1 - \frac{|pH_{input} - pH_{stored}|}{pH_{max} - pH_{min}} \quad (1)$$

For all other fields, similarity is the Symbol technique in Petrak *et al.* (1994), which is simply :

$$similarity(f_{input}, f_{stored}) = \begin{cases} 1 & if\ f_{input} = f_{stored} \\ 0 & otherwise \end{cases} \quad (2)$$

In other words, two fields either match exactly or do not match.

Note that this calculates the similarity between two fields. To calculate the actual matching, this similarity is then multiplied by the weighting of that field. This is shown in the next section on case matching.

Fields which are empty (i.e. either empty or have the value 'No Test') are handled differently here from Petrak *et al.* (1994) however. This is because, as mentioned earlier, fields with no value are not predictive of a solution. As such, empty values should not contribute to matching at all. Thus, when two fields which are empty are matched, they are not counted. As such, we have :

$$similarity([empty/No\ test], [empty/No\ test]) = No\ contribution \quad (3)$$

However, if one field has a value and the other is empty, then similarity is considered to be zero :

$$similarity([empty, No\ test], any\ non\text{-}empty\ field) = 0 \quad (4)$$

5.2.2.2 Case Similarity

To determine overall case matching, the weighted normalised sum, as used by Petrak *et al.* (1994), was used in the system, converted to a percentage. This is :

$$casesimilarity(c_a, c_b) = \frac{\sum_{f \in F} w_f \times similarity(v_{fa}, v_{fb})}{\sum_{f \in F} w_f} \times 100 \quad (5)$$

Where :

c_a and c_b are the cases being compared.

F is the set of fields being compared.

v_{fa} is the value of field f in case a

v_{fb} is the value of field f in case b

w_f is the weight of field f.

Note that the overall maximum weighting, i.e. SUM w_f is not fixed. Instead, it varies according to the cases being matched. As mentioned earlier, similarity between two fields which are empty makes no contribution to field matching. Thus, in case matching, if v_{fa} and v_{fb} are both empty, then w_f is not added to the sum, thus totally ignoring this field.

For illustration purposes, the matching between stored case 000021 (Figure 5.1), and an input problem created by using case 000642 (Figure 5.5) is shown. In Figure 5.6, the matching between them using FWNN-EXP is illustrated , while in Figure 5.7, the matching is shown using SFVW for the TREM SOLUTION. Note that in both cases, only fields where either of the cases contains data are matched. All those where for both cases, the value is empty/No test, the field does not contribute to the overall matching. Note also how in Figure 5.7 (the SFVW matching), only two

fields are used in matching between the two cases. These are the two fields in the stored case (000021 - see Figure 5.4) which were determined to be predictive for the TREM SOLUTION for that particular case, namely wstate and flamm. As such, they are the only fields used when matching an input problem (000642) against 000021, regardless of the contents of the input problem. These tables are discussed further in section 5.2.3.

	Field Name	Value
REPRESENTATION FIELDS	Sheetno	000642
	Constit	OIL; INK; WATER; THINNERS;
	Wstate	LIQUID
	Odour	WEAK
	ph	6
	Flamm	NEGATIVE
	Flash	NEGATIVE
	Hvymetalsp	NEGATIVE
	Chromesp	NEGATIVE
	Fluoridesp	NEGATIVE
	Chloridesp	NEGATIVE
	Sulphatesp	NEGATIVE

Figure 5.5 : An Input Problem using data from stored case 000642

Field Name	Field Matching	Weight	Matching after applying Weight
Constit	0.5 (2/4)	3	1.5
Wstate	1	2	2
Odour	1	2	2
ph	0.93 (2dp)	4	3.71
Flamm	1	4	4
Flash	0	4	0
Hvymetalsp	1	4	4
Chromesp	0	4	0
Fluoridesp	1	2	2
Chloridesp	1	2	2
Sulphatesp	1	2	2
	Totals	33	23.21
		Case Matching	70.35%

Figure 5.6 : Matching between stored case 000021 and input problem 000642 using Fixed Weight Nearest Neighbour using Expert Judgement (FWNN-EXP)

Field Name	Field Matching	Weight	Matching after applying Weight
Wstate	1	1	1
Flamm	1	1	1
	Totals	2	2
		Case Matching	100%

Figure 5.7 : Matching between stored case 000021 and input problem 000642 using Separate Field Variable Weight (SFVW) for the TREM SOLUTION

5.2.3 Retrieval of Cases

In terms of retrieval, all five techniques were tested making use of the nearest neighbour technique for matching and retrieval, whereby the input case is matched with every case in the case base independently. As retrieval algorithms were not in the scope of this research, it was decided to keep to this simple and complete method.

For the first two techniques (EWNN and FWNN-EXP), F, the set of fields being compared, consists of all REPRESENTATION fields, excluding those where the value is empty/No test in both the stored case an the input problem, as is seen in Figure 5.6. As such, matching is a fairly simple task of comparing all REPRESENTATION fields (except excluded ones) in the input problem and stored cases, and then computing the case matching. For FWNN-80:20, the task is hardly more difficult, as the only complication is that F consists only of the selected fields as indicated in Table 5.2 rather than all REPRESENTATION fields.

Once this matching is computed, the value (a percentage) is inserted in a list along with a reference to the stored case. This list is sorted by these matching values. Thus, once all cases have been matched against, we have a list containing the matchings of the input problem to all stored cases, sorted from highest to lowest matching.

For these first three techniques, the structure of F and, therefore, the matching of cases, is fairly simple and straightforward. For VWNN, F is slightly more complex. As mentioned, this technique only indexes on those fields which are predictive for a particular case by using the contents of the NEW (reason) fields for indexing. Therefore, in matching and retrieval, F should consist only of those fields listed in the NEW (reason) fields of a particular stored case. Figure 5.3 illustrated this for case 000021, where F would consist of all non-zero weighted fields.

Thus, when an input problem c_a (eg 000642) is matched against stored case c_b (eg 000021), F is constructed by extracting all those fields listed in the NEW (reason) fields of c_b. These fields are then all assigned a weight of one, and used for matching. Note that in this way, any field can effectively have a weight of more than one. For example (see Figure 5.3), Wstate has a weight of 2 for case 000021, as it appears in two NEW (reason) fields. Thus, for VWNN, the weight of a field is the number of times it appears as a reason.

The F thus constructed is then used for matching the input and stored cases. VWNN differs in no further way from the previous three techniques. Thus, for retrieval, the matching is calculated

between the input case and each stored case, and the result is a sorted list of matchings for the input case.

The last technique, SFVW, represents a move away from this. As described, SFVW makes use of the NEW (reason) fields contents to create multiple case-specific indexes per case. Specifically, it creates one index for each of the SOLUTION (non NEW) fields. Figure 5.7 shows the index for the TREM SOLUTION field. Naturally, the retrieval method obviously differs slightly from the other techniques, and operates as follows.

Firstly, instead of one list being created, there are six lists created, one list for each SOLUTION (non NEW) field. When matching the input case to a stored case, the following is done. Firstly, F is constructed using only the disposalrn field of the stored case. This is then used for matching against the input case. Having calculated the matching, the value and case reference are then stored in the first list - the disposal list. F is then re-created using the contents of the sitern field, and the process is performed for the site field. This process is then repeated for each of the other SOLUTION fields. In other words, each stored case is matched to the input case six times, once for each of the SOLUTION fields, and six separate lists are created. The result of this SFVW retrieval process is that we have a list of cases for each of the SOLUTION fields.

5.2.4 Discussion of Design of Indexing Techniques

In Section 5.1, the case representation was discussed with reference to the earlier chapter on case representation (Chapter 2). Here, the same strategy will be followed, thus in the discussion reference will be made to Chapter 3. In this discussion, two major areas must be covered :

- A discussion of the five techniques in relation to each other.
- An evaluation of each technique in relation to the four areas of consideration for indexing discussed in Chapter 3, namely :
  - Desirable Qualities of Indexes.
  - Indexing Vocabularies.
  - Methods for Index Assigning.
  - Memory Organisation Strategies.

Some of these sections can be discussed for all five techniques, while for others, each technique needs individual attention. In addition, it is useful for discussion purposes to consider some issues in a different order from which they appear above. As such, the structure of the discussion will not follow the order presented above. Nevertheless, it is important to bear in mind the above points when discussing the indexing techniques.

5.2.4.1 Memory Organisation Strategies

Probably the most important point of commonality amongst the indexing techniques is their memory organisation strategy. In Section 5.2.3 it was noted that all techniques make use of the Nearest Neighbour approach to search. In terms of memory organisation, this flat-memory technique whereby all cases are examined separately was referred to in Chapter 3 as Associative Memory. In this technique, there is no specific ordering of the cases in the case base. Rather, indexes are kept at a case level (i.e. no generalisation is done). While this is not always considered ideal, in reference to the specific problem there are a number of good reasons for following this approach.

If one were to examine the hierarchical type indexing of cases (described in this dissertation as a discrimination network approach to memory organisation), one would see (as discussed in Chapter 3) that the major reasons for this organisation of memory are to differentiate cases where flat differentiation is insufficient, to allow for generalisation for adaptation, and to improve speed of retrieval. Generalised indexes are derived from case-specific ones, and the result of memory organisation is a speeded up search. However, this speed is not actually resulting in better solutions, just faster ones. In fact, one of the dangers, as pointed out in Chapter 3, is the missing out of important information due to non-coverage of the entire case base.

The above points makes associative memory a good common memory organization for the techniques for a number of reasons.

Firstly, the purpose of the CBR techniques under consideration is specifically the identification of the correct cases at retrieval time. Rather than aiming towards speedy retrieval, the specific

interest was to identify the correct information. As a result of this, no attempt was made to speed up search and retrieval of cases, but instead the simplest method of organising case memory was chosen.

Secondly, this simplest method is used due to its reliability. As discussed in Chapter 3, a hierarchical-type approach can theoretically be more risky due to the fact that relevant information may be overlooked due to not all cases being accessed. In a sensitive domain such as this, where mistakes could be disastrous, it is important to ensure that incorrect or sub-optimal solutions are avoided. As such, an associative approach whereby all cases are examined appears to be an advisable memory organisation for the domain. At the very least, it is advisable to not experiment with memory organisation until such a time as a satisfactory method for labelling of cases has been ascertained. In addition, we have seen that cases contain all information necessary for indexing within themselves, thus not needing any grouping or generalisation for indexing purposes.

Thirdly, as adaptation was not being researched, there was no need for generalisation for this purpose. Lastly, keeping a common memory organisation allows us more easy analysis of the performance of the indexing techniques.

5.2.4.2 Indexing Vocabulary

In terms of an indexing vocabulary, the techniques tested can not all be grouped together. For the first group, namely those using a fixed weighting technique (Equal Weight Nearest Neighbour (EWNN), Fixed Weight Nearest Neighbour using Expert Judgement (FWNN-EXP) and Fixed Weight Nearest Neighbour using the 80-20 Rule (FWNN-80:20)), a discussion of the indexing vocabulary is fairly simplistic. Due to the fact that cases are all weighted using a global weighting scheme, there is no case-specific indexing. As such, indexes in the form of specific labels for cases do not exist. Therefore, the guidelines surrounding the indexing vocabulary are not applicable, as there are never any new indexes.

However, with Variable Weight Nearest Neighbour (VWNN) and Separate Field Variable Weight (SFVW) this is not the case. As mentioned, in these two techniques, cases are indexed individually on their own specific predictive attributes. This means that they bear evaluation with respect to the guidelines outlined in Chapter 3. All three guidelines can be discussed at the same time.

The requirements of the vocabulary are (see Chapter 3, section 3.2) :

- It should be specific enough to differentiate cases, yet general enough to cover all possible tasks.
- It should be able to handle all values in the case base.
- It should be able to handle future expansion of the case base, i.e. new cases and values.

The method of indexing cases in VWNN and SFVW solves these problems simply. As shown, indexes in these cases are contained in the NEW fields, and each of these fields simply contains field values from REPRESENTATION fields. As such, the indexing system is completely flexible.

From a specificity point of view, indexes are at a case level (i.e. not generalised). As such, differentiation between cases is simply a matter of examining values. In terms of covering all tasks, with the system this is not an issue. As mentioned, the goals in the system are overarching and well-defined and, as such, there is no need to make provision for new goals.

The handling of present and future case values, (the second and third points), is achieved simply. By using actual case values for indexes, catering for existing and new values for indexes is achieved automatically.

5.2.4.3 Theoretical Comparison of Techniques

The other two issues related to indexing, namely requirements of indexing and indexing methods, are best discussed in conjunction with a comparison of the five techniques tested. As described earlier, the indexing techniques move from a global indexing scheme to a single, case specific to

a multiple, case specific indexing. The reasons for these choices can be found by examining these last two issues related to indexing.

As stated in Kolodner (1993), shallow indexing of flat memory, (i.e. a standard nearest neighbour approach), is by far the most common means of indexing case-based systems. This type of technique is fairly easy to implement, requiring little effort and no need for complex algorithms. Due to its easy nature and the fact that it is very common, it was decided that the first technique to be tested on the system would be of this variety.

Specifically, the first three techniques mentioned, i.e. EWNN, FWNN-EXP and FWNN-80:20 are all of this nature. As mentioned, these are all using specific, fixed weighting schemes which apply a global weighting to all cases. In terms of the desirable qualities of indexes, one can see some progression amongst these techniques.

The first technique, EWNN, cannot, except in the most simple situations, be viewed as a realistically useful indexing method. By assigning equal weights to all fields, what this index is effectively saying is that no fields hold any more importance than others. As we have seen, in the domain under study this is certainly not the case. Instead, not only are certain fields more important than others, but the relative importance of fields varies from case to case. As such, EWNN cannot really be seen as particularly useful in itself as it makes no attempt to satisfy one of the most important requirements of cases, namely, identifying the predictive features of a case. Nevertheless, as the most simple indexing method, it was decided to include this technique as a basic benchmark against which others, more specific indexing methods could be measured.

As has been shown, certain fields hold more importance than others in the cases. As such, it is logical that indexing should identify the more important fields. To achieve this identification, two techniques were used, namely FWNN-EXP and FWNN-80:20. With both of these, the predictive fields in cases are identified by assigning higher weights to these fields (See Tables 5.1 & 5.2). Note that in both cases, this weighting method is still a global method of weighting.

FWNN-EXP takes into account expert opinion of which fields are most predictive, while FWNN-80:20 takes a different approach by applying a well-known heuristic to the weighting scheme. This provides an interesting contrast to expert opinion weighting, as will be seen in the results.

While FWNN-EXP and FWNN-80:20 can be viewed as complete techniques (as they are used in real world reasoners), examination of them shows a number of potential drawbacks, which led to the improvements suggested in the final techniques. Two of the major requirements of indexing are that they be predictive and useful. An examination of the case structure shows that these two techniques discussed only partially satisfy these requirements.

In terms of predictiveness, the requirement of indexes is that they identify specifically the predictive features of each case. It has been shown in the case structure that certain fields in a case are more important than others. From this perspective, both FWNN techniques do satisfy the requirements of predictiveness, as they differentiate between fields based on their relative importance. However, it was also discovered that the important fields vary from case to case. Using the global weighting method of the FWNN techniques, the same fields in all cases are always identified as being predictive. Clearly with knowledge of the case structure it can be seen that the FWNN techniques are not completely predictive for all cases.

In terms of usefulness of indexes, much the same applies. By using a global weighting scheme, and thus not identifying the predictive features of any specific case, the FWNN techniques can be viewed as not completely useful. This is because a global weighting scheme will indicate that the same set of fields are important for all cases, when in actual fact for a particular case, those may not be the important (predictive) features.

Figure 5.3, which shows weighting of case 000021 using VWNN, illustrates how different VWNN weighting can be for a particular case compared to FWNN-EXP (which is shown on Table 5.1). As can be seen in these illustrations, the global FWNN-EXP technique, while

identifying certain fields as being more important than others, does not achieve complete predictiveness or usefulness for particular cases, where the importance of fields varies.

The above discussion can be explained differently by saying that the FWNN techniques are not making use of a knowledge-based indexing strategy. In Chapter 3, knowledge-based indexing was described as the most desirable indexing method for CBR in certain circumstances due to the fact that each case is indexed on its own predictive features. It was shown how, in theory, knowledge-based indexing satisfies the requirements of case indexing best, and is thus a desirable strategy.

It has already been shown in the discussion that the predictive features vary from case to case. It has also been shown how the FWNN techniques, while identifying certain fields as being more important than others, do not cater for this variability. Thus, it can be said that a knowledge-based indexing approach was also needed for the system.

With these points in mind, the VWNN technique was developed. From the requirements of case content, it was shown how the NEW fields were added to the case structure. In addition, in Section 5.2.1.4 and Figure 5.3, we saw how these new fields were used in VWNN as complete indexes for a case. If we examine these indexes, we can see that they satisfy the requirements of predictiveness and usefulness of indexes for the system. As these NEW fields contain the actual fields in the REPRESENTATION section which led to the derivation of solutions in the SOLUTION section, they identify exactly the predictive features of any case. These fields are not influenced in any way by any type of global weighting scheme, but instead each index is totally case specific and, hence, provided the case contains correct knowledge, each case is indexed completely and only on its predictive features. In other words, we can say that knowledge-based indexing of the cases occurs.

From a usefulness point of view, much the same argument can be made. The NEW fields contain specifically the information indicating why the particular solutions were produced. In other words, specifically the information pertaining to the goals of the reasoner. Thus, as the indexes pertain only to our specific goals, we can say that they are useful.

The last technique arose due to the goals of the reasoner. If we look at a case, we see that the goal is to provide solutions for all of the SOLUTION fields. However, closer examination reveals an interesting point. A case does not have one goal (i.e. find a solution) with six fields. Rather, the case contains six separate goals. The reason why this differentiation can be made is simple. If we examine the actual contents of the NEW fields in a case (see Figure 5.1), we see that the six SOLUTION fields do not always have the same reasons for their derivation. Instead, the reasons for one SOLUTION (eg Disposal) are different to the reasons for another (eg Hazchem).

If we look at the VWNN technique, we see a potential for error. In matching and retrieval with VWNN, a single matching is produced between the input case and a stored case. This is produced by obtaining matchings over all NEW fields. However, as just pointed out, these NEW fields do not all have the same contents. Thus, what has been done in this matching is, while using all six separate indexes, they have only been used as a combined index for the overall goal of the case. The requirements of predictiveness have been satisfied, as the index is case specific, but nevertheless some detail will be lost in the matching. Specifically, an index for one field, (eg. disposalrn for disposal), will contain certain fields. This can be viewed as a predictive index specifically for disposal only. However, in VWNN, this index is combined with the other indexes. The result is that its specificity to disposal is masked by other index contents.

To overcome this masking, SFVW was developed. Here, as discussed earlier, a separate retrieval is performed for each of the six solution fields. This removes the masking effect just described. Effectively what this achieves is to have multiple, case specific indexes on each single case. In other words, each case is now indexed separately on each goal, rather than once for all goals, as in VWNN.

5.3 System Implementation

The techniques discussed were implemented on two separate platforms, one for control purposes, the other for the purposes of analysing the performance of the five techniques designed with reference to each other and other real-world reasoners.

The first platform used for implementing the techniques was using a commercially available CBR shell called ESTEEM. This was used to implement the EWNN and FWNN-80:20 techniques. Cases for the ESTEEM system were stored using ESTEEM's own internal case format. The ESTEEM implementations were not used to assess the performance of the five indexing techniques. Rather, they were used as controls. The presumption was made that as a commercial CBR tool, ESTEEM's indexing and retrieval would function correctly. As such, the ESTEEM implementations were used to provide figures against which the custom implementations could be compared. If the custom implementations behaved similarly to the ESTEEM implementations, this would be taken as a validation that the custom implementations were behaving as they should.

The second platform was one custom built by the author implementing the techniques from scratch using programming to develop CBR modules. Thus, all five techniques were programmed using Delphi 1.0. These were all incorporated into a single program called :
HACA - The **Ha**zardous **C**hemicals **A**dvisor.

Cases for HACA are stored in a single Paradox table, one record per case, and all coding was done in Object Pascal, Delphi's coding language. All five routines are available in one program, thus allowing all five techniques to be tested on the same input case. The purpose of these custom implementations was to assess the performance of the five techniques with reference to each other and to other CBR systems.

Both systems were developed and tested under Windows 95, running on a 486 DX4/100 PCI IBM PC compatible with 48MB of RAM.

5.4 Conclusion to Chapter 5

In this chapter, the techniques implemented and tested in the research have been examined. As identification of correct cases was the main interest of the CBR research, the research concentrated on indexing techniques. As indexing is dependent on case representation, a single case representation was defined based on the principles of case representation as discussed in Chapter

2. Following this, five indexing techniques were implemented. All using a flat memory organisation with nearest neighbour search for consistency purposes, the indexing techniques represent an evolution from a single global indexing for cases to a single case-specific indexing to multiple case-specific indexing for separate goals. Two techniques were implemented in a commercial CBR shell, ESTEEM, as a benchmark for the custom implementation. All techniques were then implemented in custom CBR modules for the purposes of evaluating the five techniques' performance.

6. EXPERIMENTS AND RESULTS

6.1 Formulation of Experiments

6.1.1 Testing Methods used in CBR research

O'Leary (1993) reported that validation of a case-based system generally involves comparing system outputs to human experts or machine solutions. As actual data sheets are used for case representation, access to human expertise is readily available for testing purposes in the form of the actual solutions listed on the sheets.

In the literature, testing of case-based reasoners, much like other issues such as representation and indexing, tends to be highly application specific. Cohen (1989) commented on the lack of methodology observed in CBR research. Thus, as with indexing and representation, the literature survey did not identify a specific, well-defined testing method to follow in this study. In addition, there was very little in the literature on the principles behind actual testing of systems. As such, it was decided to base the testing on the tests used in other case-based reasoners.

Testing, according to the literature, generally takes the form of running a number of test cases through a case base and obtaining a number of results. Typically, the basic issues to assess are :

- What size should the case base used for tests be?
- How many input cases should the system be tested on?
- What results should be captured from the tests to assess performance of the system?

However, within these parameters, a wide degree of variability occurs. In terms of case base size and the number of input cases, the numbers vary widely from system to system. In terms of results captured, the information needed again varies from application to application. Some examples are :

- CARMA (Branting & Hastings, 1994) makes use of eight, 15 and 48 cases for case base size, and in its results assesses mainly issues related to error.

- OBLIVION (Langley & Sage, 1994) uses 20 case bases of 200 cases with 100 test cases, and measures classification accuracy.

- MetVUW (Jones & Roydhouse, 1994) used a 550 case base, used five input cases, and tested precision and recall.

- CLAVIER (Hennessy & Hinkle, 1992) started with 30 cases, moving to 150 at publishing, and was tested on site.

- ARCHIE (Pearce *et al.*, 1992) used 20 cases in the case base.

- CASCADE (Simoudis, 1992) used a case base of 141 cases, with 136 of these as input cases, and examined precision and recall.

- Petrak *et al.* (1994) converted two available databases to case bases of 547 and 288 cases in size. For testing, the case bases were divided into ten equal-sizes sets, and testing was performed on each set using the other nine sets as a case base. Error rate and Output similarity were then gathered as results.

- For Battle Planner (Goodman, 1989), it is unclear whether 605 or 145 cases were used for the case base. For testing, 10% of the case base was used for input cases, while the rest were used as the actual case base for retrieval. Accuracy was the main result gathered, but retrieval time was also discussed.

- The work of Simoudis & Miller (1990) on validated retrieval used a case base of 200 cases, and examined selectivity of cases as a percentage.

- Buta (1994) used case libraries of 1039 and 2143 cases, with 10% as input and the other 90% as a case base. Performance was measured by comparing system results with real-world ones.

The approaches of these examples, in conjunction with the goals of the experiments (section 6.1.2) were used to determine the experiments to perform. The discussion of the actual experiments is found in section 6.1.3.

6.1.2 Goals of the experiments

Experiments were designed with four goals in mind, namely :

- Comparison of HACA's implementation of indexing techniques with ESTEEM's for the purposes of validating the HACA implementations.
- Comparison of the five indexing techniques as implemented in HACA to determine which of the proposed indexing techniques performed best.
- Comparison of the performance of techniques with other implementations to determine whether any of the techniques performed in a comparable manner.
- Decide what implications the results have for building of an intelligent decision support system using CBR.

The purpose of implementing certain techniques in ESTEEM as well as HACA was to use ESTEEM as a validation mechanism for the custom implementation. The presumption made was that if the ESTEEM and HACA implementations of techniques produced results which were comparable, it could be concluded that the HACA techniques were behaving as they should. This would then allow confident comparison of the five techniques as implemented in HACA. As such, the two ESTEEM implementations (EWNN-EST and FWNN-80:20-EST) were compared to the HACA implementations (EWNN and FWNN-80:20) of the same techniques.

The ESTEEM implementations were never intended for full analysis. It was decided that actual technique performance would be evaluated by comparing the performance of the five techniques as implemented in HACA. The performance of the five HACA implementations, namely EWNN, FWNN-EXP, FWNN-80:20, VWNN and SFVW, were thus compared. In this case, the primary goal in analysing the results was to determine which of the five indexing techniques could be considered to be performing best.

While it could be decided which of the five techniques performed best, there would be no means of measuring whether this best technique could be considered satisfactory unless there was some means for external validation. Results were therefore compared with relevant results from other implementations. This allowed assessment of the value of the techniques, and also served to highlight areas of potential improvement.

Finally, and largely in conjunction with the second and third goals, the implications of the results were assessed with reference to using the techniques for a real world implementation.

6.1.3 Description of experiments performed and results obtained

From the testing methods of other reasoners described in section 6.1.1 and the goals of the experiments stated in section 6.1.2, it was decided to use an approach similar to that followed in Battle Planner, Petrak *el al* (1994) or Buta (1994) to test the techniques. This involves using actual cases stored in the case base as inputs to the system, determining solutions based on retrieved cases, and then gathering results deemed relevant (for the actual method, see section 6.1.4).

In section 6.1.2, the goals of the experiments hinge on the issue of assessing performance. As the focus of the research is on indexing, it is essential to choose the correct means to assess performance of indexing techniques. As from an implementation point of view, indexing's goal is the location of the most relevant cases at retrieval time, an experiment which allows assessing of this retrieval of relevant cases is needed. It was decided that prediction accuracy, as used in Battle Planner (Goodman, 1989) or Petrak *et al* (1994) (who use the reciprocal of this, and call it error rate), would be used to assess this performance (see section 6.1.4.3 for the method used to calculate prediction accuracy).

The premise used is that for two techniques (A and B), if technique A produces more accurate solutions than technique B, then it can be concluded that technique A does a better job of locating the relevant cases, and hence is a better indexing method (all other factors being the same). Based

on this, the prediction accuracies of the five techniques in HACA, and the two in ESTEEM, were gathered and compared using statistical techniques to evaluate relative performance.

The statistical technique used to compare prediction accuracies was the paired *t*-test with a 5 percent level of significance as described in Pollard (1977). The paired *t*-test is used to compare two sets of observations where the data is paired. For comparing prediction accuracies in this research, the data is paired, as results are obtained for the same input cases over all techniques tested. (See section 6.1.4.4 for testing method).

In order to test whether two sets of observations can be considered not significantly different, the *t*-test is performed with a null hypothesis of equality :

$H_0 : \mu = \mu_0$

$H_1 : \mu \neq \mu_0$

where

μ is the difference between the means of the two sets of observations.

$\mu_0 = 0$.

The critical region for equality with a 5 percent level of significance is the upper 2.5 and lower 2.5 percent of the *t*-distribution with n-1 degrees of freedom, where n is the number of pairs in the observation sets. The upper 2.5 percent region can be used alone due to the symmetry of the *t*-distribution.

In order to test whether one set of observations (called set B) is significantly higher than another (called set A), the *t*-test is performed with a null hypothesis of inequality, where the differences between the means is calculated by subtracting values in set A from values in set B :

$H_0 : \mu \leq \mu_0$

$H_1 : \mu > \mu_0$

where

μ is the difference between the means of the two sets of observations.

$\mu_0 = 0$.

The critical region for this inequality with a 5 percent level of significance is the upper 5 percent of the *t*-distribution with n-1 degrees of freedom, where n is the number of pairs in the observation sets.

If the computed test statistic falls within the critical region, then we reject the null hypothesis, otherwise we accept it.

Retrieval time of cases was also gathered. This data was used for three purposes. Firstly, it was used in comparing the HACA implementation of techniques with the ESTEEM implementation. This was done to ascertain that the techniques in HACA did not carry unnecessary overhead in implementation. Secondly it was used to determine whether any of the five techniques as implemented in HACA caused a degradation in retrieval speed compared to others. Thirdly, the times were used to assess whether the techniques tested could realistically be implemented in a real-world situation. As retrieval time was not the primary data required, it was not analysed statistically.

Matching figures of cases retrieved were gathered, along with the number of cases retrieved. This was used in conjunction with prediction accuracy values to assess the certainty with which different techniques retrieved cases. The reasoning for the use of this data was as follows.

If a single technique is taken in isolation, it can reasonably be concluded that if one case (case A) is retrieved with a matching of 60%, while another (case B) is retrieved with a matching of 40% (i.e. lower than A's matching), then in terms of the functionality of that technique, case A is considered a better candidate for solution construction than case B. When one compares the matching figures of two techniques, as is done in this research, the question which has to be addressed is - Do higher matching figures in one technique indicate that the technique is "more certain" of the usefulness of it's retrieved cases? As will be seen, this can be assessed in conjunction with number of cases retrieved and prediction accuracy.

6.1.4 Experiment Methods

Having described the experimental data gathered and the purposes for gathering it, the actual method used for performing the experiments will be described. Before this, describing the actual method, some points need to be discussed.

6.1.4.1 Restrictions on the Case Base

As can be seen from the short list in section 6.1.1, there is great variability amongst reasoners in terms of all three testing issues. In particular, results gathered are highly domain dependent. Also, case base size varies according to availability of cases. For instance, most reasoners with large case bases either have generated cases or else pre-existing databases to draw on. A system such as ARCHIE which started off with only 20 cases illustrates that often it is not so easy to readily obtain real-world cases. While in the domain under investigation, there are many completed data sheets, using them all presents two problems :

- They are not stored on computer, thus all data sheets would have to be entered by hand. This in itself would not be a limiting factor up to a point. However,
- As mentioned, certain fields had to be added to the case representation, specifically those linking the REPRESENTATION fields and SOLUTION fields. In order to obtain this data, expert knowledge was required. However, expert time is valuable, and thus greatly limited the number of data sheets actually available.

As a result, a case base of ninety cases was constructed.

6.1.4.2 Solution Construction

As HACA's intended role is as a decision support tool (as in Kolodner, 1991), and not an expert system, no attempt is made to build solutions automatically. However, while in a purely decision role, HACA would assist an expert, for testing purposes, there is a need to actually construct solutions using retrieved cases. Thus, to allow calculation of prediction accuracy, for the first four techniques, the top ten cases retrieved were taken, and an attempt was made to build solutions for the input case based only on the data contained in the retrieved cases. For SFVW, for each

SOLUTION field of each case, the top three cases retrieved were taken and used to construct a solution for that SOLUTION field.

Note once again that including the input case in the case base does not affect the reasoning process, as this case (if retrieved) is always ignored for solution construction purposes.

6.1.4.3 Prediction Accuracy

While retrieval time and matching figures are obvious, calculation of prediction accuracy warrants some explanation. As there are a number of solution fields in a case and, possibly, a number of components to each solution, it is possible to make some correct and some incorrect predictions for the same case. Thus, prediction accuracy is calculated as :

$$\frac{\textit{Total number of solutions} - \textit{Total number of incorrect guesses}}{\textit{Total number of solutions}} \qquad (6)$$

Where :

Total number of solutions = Maximum of the total number of solution components of the input case and currently matched stored case. For SFVW, this total refers to a field, while for all other techniques, this refers to all SOLUTION fields.

6.1.4.4 Description of the Testing Method

Having discussed the methodology followed for experiments, the actual testing method used was as follows :

- Twenty cases were chosen at random from the case base. This same set was used for tests on all five techniques.
- Each case was used as an input to the system but, unlike in many other systems, this case was not removed from the case base. The reasoning for this was that for retrieval, the objective was to examine whether an exact match present in the case base would be

retrieved accurately. Note that unlike in a reasoner such as Battle Planner, where indexing is done over the whole case base by comparing cases, the system designed here strictly indexes at the case level (even with its global indexing), thus including the "ideal" case in the case base does not affect indexing. Obviously, this ideal case was not used for solution construction.

- Retrieval using each technique was made on the current input case, using a threshold matching level of 0.3 (30%) for retrieval.
- For the first four techniques (Equal Weight Nearest Neighbour (EWNN), Fixed Weight Nearest Neighbour using Expert Judgement (FWNN-EXP), Fixed Weight Nearest Neighbour using the 80-20 rule (FWNN-80:20) and Variable Weight Nearest Neighbour (VWNN)), the top ten cases of those retrieved (or all cases when 10 or less were retrieved) were taken, and used to construct a solution for the input case.
- For Separate Field Variable Weight (SFVW), the top three cases for each SOLUTION field were used to construct a solution for that field.
- The following data were then calculated for each input case for the first four techniques :
  - The time taken for case retrieval. These results are shown in Table 6.1.
  - The average matching of the cases retrieved for each case retrieval. These results are shown in Table 6.2.
  - The average matching of the top ten (or less) cases retrieved for each case retrieval (i.e. the cases used for solution construction). These results are shown in Table 6.3.
  - The number of cases retrieved. These results are shown in Table 6.4.
  - The prediction accuracy of each retrieval. These results are shown in Table 6.9.
- For SFVW, due to its different retrieval nature, the following results were gathered :
  - The time taken for case retrieval. These results are shown in Table 6.1.
  - The average matching of the cases retrieved for each SOLUTION field. These results are shown in Table 6.5.
  - The average matching of the top 3 (or less) cases retrieved for each SOLUTION field. These results are shown in Table 6.6.
  - The number of cases retrieved for each SOLUTION field. These results are shown in Table 6.7.

- The prediction accuracy of each retrieval for :
  - Each SOLUTION field in each case. These results are shown in Table 6.8.
  - Each case. These results are shown in Table 6.9.
- To evaluate performance, prediction accuracies of the techniques were compared using paired *t*-tests as described in Pollard (1977) and discussed in section 6.1.3.
  - In order to determine whether the custom implemented techniques (i.e. the HACA implementation) were operating as they should, FWNN-80:20 and FWNN-80:20-EST prediction accuracies were compared with a null hypothesis of equality (see section 6.1.3). The result of this comparison is shown in Table 6.10. (EWNN and EWNN-80:20 prediction accuracies could not be compared. See section 6.3.1 for details).
  - In order to determine the relative performance of the five techniques implemented in HACA, each techniques prediction accuracy was compared to the prediction accuracy of each technique which returned a lower average prediction accuracy. Comparisons were performed using a null hypothesis of inequality (as described in section 6.13) to determine whether the technique with the higher average prediction accuracy could be considered superior to that with the lower prediction accuracy. The results of these comparisons are shown in Table 6.11.

6.2 Results

The results obtained from the experiments are presented in Tables 6.1 to 6.11. Table 6.1 shows the time taken to search the case memory of 90 cases. The two techniques implemented in ESTEEM are listed first. The retrieval time for each case in each technique is given (rounded to 1dp), and an average of the retrieval time for all ten cases is given for each technique.

Table 6.1 : Retrieval Time for Input Cases and Indexing Techniques in Seconds

Input Case Code	EWNN - EST	FWNN - 80:20 - EST	EWNN	FWNN - EXP	FWNN - 80:20	VWNN	SFVW
000086	16.4	5.5	3.1	3.1	3.2	3.1	3.3
000115	16.5	5.4	3.1	3.1	3.2	3.2	3.1
000292	16.5	5.8	3.1	3.1	3.2	3.1	3.1
000347	16.5	5.9	2.9	3.1	3.2	3.2	3.1
000366	16.6	6.1	3.1	3.1	3.2	3.2	3.2
000642	16.7	6.8	3.3	3.3	3.3	3.3	3.1
005164	16.6	6.9	3.1	3.1	3.2	3.3	3.1
005269	16.8	6.3	3.2	3.1	3.3	3.3	3.2
007844	16.8	5.9	3.1	3.1	3.2	3.3	3.1
100411	16.8	6.4	3.1	3.1	3.2	3.3	3.1
000280	16.7	6.8	3.3	3.2	3.2	3.2	3.2
007868	16.7	5.9	3.3	3.3	3.3	3.3	3.3
000862	16.9	5.7	3.2	3.3	3.3	3.2	3.3
000966	16.7	6.8	3.2	3.2	3.2	3.1	3.3
007091	16.6	7.1	3.2	3.2	3.2	3.3	3.3
100026	16.6	6.3	3.2	3.2	3.2	3.2	3.3
000030	16.6	6.6	3.3	3.2	3.2	3.2	3.3
000982	16.7	6.5	3.3	3.3	3.3	3.4	3.2
007808	16.6	6.6	3.3	3.4	3.4	3.4	3.2
003669	16.7	5.8	3.3	3.3	3.3	3.3	3.3
Average	16.7	6.3	3.2	3.2	3.2	3.2	3.2

Table 6.2 presents matching figures for all cases retrieved above the 0.3 (30%) threshold for retrieval as a percentage. The matching obtained for each case for each technique is given (rounded to one decimal place), as well as an average matching for all ten cases for each technique. Note that no figures are given for the ESTEEM technique EWNN-EST, as no cases were retrieved above the threshold (thus EWNN-EST only appears in Table 6.1).

Table 6.2 : Average Matchings of all Cases Retrieved Above Threshold for Input Cases and Indexing Techniques as a Percentage

Input Case Code	FWNN - 80:20 - EST	EWNN	FWNN - EXP	FWNN - 80:20	VWNN
000086	32.0	32.3	33.0	35.3	55.8
000115	31.6	45.0	45.3	44.8	47.0
000292	35.1	43.5	46.2	45.6	45.5
000347	40.2	39.8	42.7	43.9	58.1
000366	35.1	31.8	35.2	37.4	67.6
000642	39.4	50.1	50.7	54.5	65.9
005164	41.1	47.0	47.0	47.3	65.7
005269	41.8	39.5	42.0	44.8	60.6
007844	35.1	39.4	40.1	43.6	52.9
100411	41.5	43.8	44.2	48.1	62.2
000280	46.2	51.6	53.4	54.4	67.3
007868	40.2	41.3	40.8	46.7	67.8
000862	35.5	40.2	40.0	46.3	68.0
000966	46.4	42.4	43.1	46.0	59.9
007091	38.3	38.0	40.1	45.7	68.3
100026	46.3	41.2	43.4	44.9	62.1
000030	45.5	47.0	48.3	50.5	65.8
000982	42.2	38.8	41.0	43.8	40.6
007808	45.6	40.2	44.4	49.9	46.6
003669	31.3	34.3	34.8	37.0	51.4
Average	39.5	41.4	42.8	45.5	59.0

Table 6.3 presents matching figures in the same format as Table 6.2. The difference is that these figures were calculated using the top 10 (or less) retrieved cases (i.e. the cases used for solution construction).

Table 6.3 : Average Matchings of Top 10 Cases Retrieved for Input Cases and Indexing Techniques as a Percentage

Input Case Code	FWNN - 80:20 - EST	EWNN	FWNN - EXP	FWNN - 80:20	VWNN
000086	32.0	32.3	33.6	39.0	75.8
000115	35.6	59.3	62.4	58.3	68.4
000292	38.9	55.6	62.4	59.8	64.8
000347	52.7	52.2	58.9	64.2	79.8
000366	38.4	31.8	36.6	47.0	100.0
000642	52.3	72.2	72.3	80.4	100.0
005164	50.4	66.7	68.0	64.6	100.0
005269	52.2	49.0	57.3	66.3	83.5
007844	38.1	47.1	51.3	61.0	80.1
100411	61.6	64.2	69.1	72.9	91.3
000280	64.7	75.8	77.8	80.5	100
007868	53.9	53.2	53.9	66.4	100
000862	46.4	50.0	53.1	66.3	100
000966	69.9	58.5	62.2	67.7	84.4
007091	52.7	48.4	53.6	64.5	100
100026	72.3	58.9	64.6	68.4	86.5
000030	60.8	61.4	65.7	70.4	100
000982	53.7	47.4	52.5	57.3	55.1
007808	62.4	52.5	58.8	71.1	69.5
003669	31.3	34.3	35.6	40.3	66.5
Average	51.0	53.5	57.5	63.3	85.3

Table 6.4 gives the number of cases retrieved above the threshold for each case for each technique, as well as an average number of cases retrieved for each technique. Note that for all tables up to Table 6.4, the fifth technique, SFVW, is not included in the results. This is due to the fact that it retrieves for six separate goals.

Table 6.4 : Number of Cases Retrieved above Threshold for Input Cases and Techniques

Input Case Code	FWNN - 80:20 - EST	EWNN	FWNN - EXP	FWNN - 80:20	VWNN
000086	1	4	12	24	40
000115	34	53	58	53	28
000292	37	51	57	52	27
000347	36	51	51	53	58
000366	18	6	14	38	48
000642	59	76	77	74	59
005164	54	65	68	66	45
005269	34	29	37	53	58
007844	16	38	46	55	34
100411	50	70	70	61	59
000280	72	71	70	72	55
007868	54	60	59	60	54
000862	44	31	41	57	55
000966	67	66	68	67	59
007091	73	51	59	65	57
100026	60	57	57	55	57
000030	63	64	63	63	48
000982	58	43	52	56	30
007808	62	55	50	68	41
003669	4	9	12	18	19
Average	44.8	47.5	51.1	55.5	46.6

Tables 6.5 to 6.8 contain the experimental data for SFVW (other than accuracy listed in Table 6.9, and retrieval time shown in Table 6.1). As SFVW retrieves for six separate goals for each case,

overall matching figures or number of cases retrieved are not given. Instead, separate results for each solution field are presented.

Table 6.5 gives matching figures for all cases retrieved above threshold for each of the solution fields for each case, along with average matching of all cases for each solution field.

Table 6.5 : Average Matchings of all Cases Retrieved Above Threshold for Separate SOLUTION Fields for SFVW Search

Input Case Code	Disposal	Site	Hazchem	Trem	Warning	Clothing
000086	55.9	0.0	80.7	51.8	96.43	0.0
000115	54.6	0.0	84.2	52.3	73.2	0.0
000292	56.1	0.0	85.7	53.7	66.1	0.0
000347	56.9	0.0	94.1	57.4	57.1	0.0
000366	57.3	0.0	78.9	76.1	57.1	0.0
000642	61.2	0.0	78.6	74.2	64.3	0.0
005164	53.0	0.0	78.5	72.4	57.1	0.0
005269	62.1	0.0	94.2	57.5	82.1	100.0
007844	68.1	0.0	88.7	54.6	87.5	0.0
100411	66.4	0.0	94.0	62.5	57.1	0.0
000280	57.2	0.0	78.5	75.7	41.1	0.0
007868	66.1	0.0	81.5	72.2	82.1	0.0
000862	66.4	0.0	80.8	72.3	98.2	0.0
000966	58.1	0.0	93.9	55.9	35.7	0.0
007091	72.0	0.0	78.5	70.5	60.7	100.0
100026	56.5	0.0	94.0	60.5	41.1	0.0
000030	48.9	0.0	78.3	70.6	57.1	100.0
000982	51.8	0.0	69.6	47.7	64.3	100.0
007808	47.2	0.0	87.1	61.7	41.1	0.0
003669	66.8	0.0	84.4	56.5	67.9	0.0
All Cases	59.1	0.0	84.2	62.8	64.4	20.0

Table 6.6 contains the same data as Table 6.5, except it is for the top 3 (or less) retrieved cases (i.e. the cases used for solution construction).

Table 6.6 : Average Matchings of Top 3 Cases Retrieved for Separate SOLUTION Fields for SFVW Search

Input Case Code	Disposal	Site	Hazchem	Trem	Warning	Clothing
000086	100.0	0.0	100.0	100.0	96.4	0.0
000115	97.6	0.0	100.0	100.0	73.2	0.0
000292	73.8	0.0	100.0	100.0	66.1	0.0
000347	100.0	0.0	100.0	100.0	57.1	0.0
000366	95.2	0.0	100.0	100.0	57.1	0.0
000642	100.0	0.0	100.0	100.0	64.3	0.0
005164	83.3	0.0	100.0	100.0	57.1	0.0
005269	100.0	0.0	100.0	100.0	82.1	100.0
007844	100.0	0.0	100.0	100.0	87.5	0.0
100411	100.0	0.0	100.0	100.0	57.1	0.0
000280	100.0	0.0	100.0	100.0	41.1	0.0
007868	100.0	0.0	100.0	100.0	82.1	0.0
000862	100.0	0.0	100.0	100.0	98.2	0.0
000966	100.0	0.0	100.0	100.0	35.7	0.0
007091	100.0	0.0	100.0	100.0	60.7	100.0
100026	100.0	0.0	100.0	100.0	41.1	0.0
000030	100.0	0.0	100.0	100.0	57.1	100.0
000982	100.0	0.0	69.6	55.6	64.3	100.0
007808	83.9	0.0	100.0	100.0	41.1	0.0
003669	100.0	0.0	100.0	66.7	67.9	0.0
All Cases	96.7	0.0	98.5	96.1	60.3	20.0

Table 6.7 gives the number of cases retrieved above threshold for each solution field for each case, along with average number retrieved for each solution field.

Table 6.7 : Number of Cases Retrieved Above Threshold for Separate SOLUTION Fields for SFVW Search

Input Case Code	Disposal	Site	Hazchem	Trem	Warning	Clothing
000086	23	0	43	41	2	0
000115	25	0	8	43	2	0
000292	10	0	9	43	2	0
000347	35	0	43	43	1	0
000366	17	0	43	43	2	0
000642	38	0	43	43	1	0
005164	20	0	43	43	1	0
005269	42	0	43	43	2	1
007844	34	0	9	43	2	0
100411	36	0	43	43	1	0
000280	40	0	54	61	2	0
007868	35	0	54	61	2	0
000862	36	0	54	61	2	0
000966	44	0	58	55	2	0
007091	40	0	54	61	3	1
100026	46	0	58	56	2	0
000030	31	0	54	61	3	1
000982	33	0	2	51	3	1
007808	29	0	14	55	2	0
003669	39	0	8	7	3	1
All Cases	32.7	0.0	36.9	47.9	2.0	0.3

Table 6.8 gives the solution accuracy for each solution field for each case, as well as the average accuracy for each solution field.

Table 6.8 : Accuracy of Solutions Produced for Separate SOLUTION Fields for SFVW Search

Input Case Code	Disposal	Site	Hazchem	Trem	Warning	Clothing
000086	1.00	1.00	0.00	0.00	1.00	1.00
000115	1.00	1.00	1.00	1.00	1.00	1.00
000292	1.00	1.00	1.00	1.00	1.00	1.00
000347	1.00	1.00	1.00	0.00	1.00	1.00
000366	1.00	1.00	1.00	1.00	1.00	1.00
000642	1.00	1.00	1.00	1.00	1.00	1.00
005164	1.00	1.00	1.00	1.00	1.00	1.00
005269	1.00	1.00	1.00	0.00	0.50	0.50
007844	1.00	1.00	0.00	0.00	1.00	1.00
100411	1.00	1.00	1.00	1.00	1.00	1.00
000280	1.00	1.00	1.00	1.00	1.00	1.00
007868	1.00	1.00	0.00	1.00	1.00	1.00
000862	1.00	1.00	1.00	1.00	1.00	1.00
000966	1.00	1.00	1.00	0.00	1.00	1.00
007091	0.25	1.00	1.00	1.00	1.00	1.00
100026	1.00	1.00	0.0	1.00	1.00	1.00
000030	0.33	1.00	1.00	1.00	1.00	1.00
000982	1.00	1.00	1.00	1.00	1.00	1.00
007808	1.00	1.00	0.0	1.00	1.00	1.00
003669	1.00	1.00	1.00	1.00	1.00	1.00
All Cases	0.93	1.00	0.75	0.75	0.98	0.98

Table 6.9 presents the accuracy of solutions produced for all techniques, including SFVW. The accuracy of solution for each case for each technique is given, along with the average accuracy of each technique.

Table 6.9 : Accuracy of Solutions Produced for Input Cases and Techniques

Input Case Code	FWNN - 80:20 - EST	EWNN	FWNN - EXP	FWNN - 80:20	VWNN	SFVW
000086	0.50	0.63	0.63	0.63	0.63	0.75
000115	1.00	0.50	0.83	1.00	1.00	1.00
000292	1.00	0.67	1.00	1.00	1.00	1.00
000347	0.86	0.83	0.83	0.83	0.83	0.83
000366	0.86	0.83	0.83	0.83	0.83	1.00
000642	1.00	1.00	1.00	1.00	1.00	1.00
005164	1.00	1.00	1.00	1.00	1.00	1.00
005269	0.67	0.67	0.67	0.67	0.67	0.67
007844	0.67	0.67	0.67	0.67	0.67	0.67
100411	1.00	1.00	1.00	1.00	1.00	1.00
000280	1.00	1.00	1.00	1.00	1.00	1.00
007868	0.83	0.83	0.83	0.83	0.83	0.83
000862	1.00	0.67	0.71	1.00	1.00	1.00
000966	1.00	1.00	1.00	1.00	1.00	0.86
007091	0.78	0.70	0.78	0.78	0.78	0.67
100026	1.00	0.88	1.00	1.00	1.00	0.86
000030	0.67	0.67	0.67	0.67	0.71	0.75
000982	1.00	0.63	0.63	0.75	1.00	1.00
007808	0.67	0.67	0.67	0.67	0.83	0.83
003669	0.67	0.67	0.67	0.67	0.67	1.00
Average	0.86	0.78	0.82	0.85	0.87	0.89

For the purposes of comparing the performance of FWNN-80:20 (a HACA implementation) to FWNN-80:20-EST (an ESTEEM implementation), Table 6.10 shows the result of the paired *t*-test performed using the data from the two prediction accuracies (rounded to 3 decimal places).

Table 6.10 -Results of *t*-test comparison for ESTEEM vs HACA

	FWNN - 80:20-EST
FWNN - 80:20	0.622

Upper 2.5 percent critical region of the *t*-distribution with 19 (i.e. 20 -1) degrees of freedom :

$t_{19} > \mathbf{2.093}$

$H_0 : \mu = 0$

$H_1 : \mu \neq 0$

For the purposes of comparing the performance of the five HACA implementations, Table 6.11 shows *t*-test results to determine whether a particular techniques prediction accuracy can be considered higher than another (rounded to 3 decimal places). The table should be read as follows:

- In the first row, the techniques which are being tested for superiority are listed.
- In the first column, the techniques against which the techniques in the first row were tested are listed.
- Cells in the body of the table contain the computed test statistic of the *t*-tests performed.
- Cells blacked out indicate that superiority was not tested for that column/row combination.

Eg The test statistic computed for testing whether VWNN prediction accuracy can be considered higher than EWNN is found in the cell with the column heading of **VWNN**, and the row heading of **EWNN** (and is 2.754).

Note that to test for superiority, the method used is the test for inequality as described in section 6.1.3. In this case, the values of the technique with the lower average (i.e. the technique in the first column) are subtracted from the values of the technique with the higher average (i.e. the technique

in the corresponding first row) in calculating the test statistic. If the test statistic is significant (as in the example in the preceding paragraph), the null hypothesis can be rejected, and it can be concluded that the technique being tested for superiority is superior.

Table 6.11 - Results of *t*-test comparisons for five HACA techniques

Input Case Code	EWNN	FWNN - EXP	FWNN - 80:20	VWNN	SFVW
EWNN		1.963	2.300	2.754	2.846
FWNN - EXP			1.700	2.120	2.001
FWNN - 80:20				1.555	1.345
VWNN					0.586
SFVW					

Upper 5 percent critical region of the t-distribution with 19 (i.e. 20 -1) degrees of freedom :

$t_{19} > \mathbf{1.729}$

$H_0 : \mu \leq 0$

$H_1 : \mu > 0$

6.3 Discussion of Results

6.3.1 Comparison Between ESTEEM Implementations and Custom Implementations

The purpose of the ESTEEM implementations was to validate the HACA implementation of techniques. As will be seen, comparisons gave encouraging results.

EWNN-EST did not produce any matchings above the required 30% matching precision, and thus no data could be calculated for EWNN-EST other than retrieval time. As such, comparison between HACA and ESTEEM had to be done with the FWNN-80:20 data.

The result of the paired *t*-test comparing the prediction accuracies is shown in Table 6.10. The upper 2.5 percent critical region of the t_{19} distribution is greater than 2.093. The calculated test statistic is t=0.622. This value is not significant, therefore we accept the null hypothesis and conclude that the prediction accuracies of FWNN-80:20-EST and FWNN-80:20 are not significantly different with a level of confidence of 95%.

From this result, it was concluded that FWNN-80:20 was performing as it should from an indexing perspective. This permitted comparison of the five HACA techniques in the knowledge that they were functioning as expected.

An examination of matchings (Table 6.2 and Table 6.3) for FWNN-80:20-EST compared to matchings for FWNN-80:20 shows that matchings produced by ESTEEM were generally lower than for the same technique implemented in HACA. Viewed in isolation, this might indicate that the HACA technique was more certain of the usefulness of cases retrieved than the ESTEEM implementation, which would appear to contradict the findings on prediction accuracy. However, examination of the number of cases retrieved (Table 6.4) shows a possible explanation. Just as FWNN-80:20-EST achieved lower matchings than FWNN-80:20, so the number of cases retrieved above the 30% threshold was lower in FWNN-80:20-EST. What this indicates is that rather than FWNN-80:20 actually having more certainty in its matchings (hence higher matching figures) than FWNN-80:20-EST, it simply gave higher matching figures generally, which in turn led to more cases being retrieved above threshold..

It is felt that this lower matching and number of cases retrieved was due to the difference in the way in which ESTEEM handles blank entries to HACA's handling of NO TEST/blank cells (as discussed earlier, HACA ignores these fields, rather than letting them detrimentally affect the matching figure).

Retrieval time (Table 6.1) shows results which are encouraging for HACA, but also hold indications for some possible improvements. As can be seen, for EWNN, HACA's performance greatly exceeded the performance of ESTEEM for equal size case bases. However, a possible area of improvement in HACA's performance is nevertheless highlighted. In all four techniques tested in HACA, retrieval time was the same on average (when rounded to 1dp). However, in ESTEEM, retrieval time dropped from EWNN-EST to EWNN-80:20-EST by 62% (rounded off). In EWNN-80:20-EST, matching was performed on far fewer fields than in EWNN-EST. Thus, the drop in retrieval time was interpreted as a result of less fields being matched. HACA does not achieve this improvement. There is, thus, the opportunity to improve the retrieval time of HACA. Through more efficient programming.

6.3.2 Discussion of the Five Techniques as Implemented in HACA

Having established the comparability of HACA's indexing techniques to the same technique in ESTEEM, and thus verified the accuracy of HACA's implementation of techniques, an analysis of the performance of the indexing techniques designed will be given by analysing the results of the HACA implementations. As mentioned, the results were analysed with the following intentions in mind :

- Comparison of the five indexing techniques to assess which was best (from an indexing point of view).
- Comparison of the results to those of other implementations to assess if the techniques were effective.
- Evaluate the implications of the results on the building of a complete intelligent decision support system using CBR as a basis.

When analysing the data, it will be shown that improvements might be achieved in certain areas. However, in such situations, it is important to remember the focus of the research. The CBR research was focussed specifically on indexing, which from an implementation point of view is a focus on ensuring that the most relevant cases are retrieved. The result of this is that it would

be expected that the potential for improvement would be found in other areas. However, these are separate research areas which were not the focus of this dissertation. Thus, when improvements are identified, this does not invalidate the results. Rather, it simply indicates more areas of potential research in this domain.

6.3.2.1 Prediction Accuracy

As discussed, prediction accuracy was used as the test of effectiveness of the techniques, with other data providing backup and/or insight into future research issues. An examination of the accuracy of solutions produced from the five techniques for indexing (Table 6.9) indicates an increase in average values from EWNN to FWNN-EXP to FWNN-80:20 to VWNN to SFVW (Note that FWNN-80:20-EST is not included in comparisons). These results were compared statistically using paired *t*-tests with a null hypothesis of inequality to determine whether any particular technique which returned a higher average prediction accuracy could be considered significantly higher, and hence better, than a technique with a lower accuracy.

The test statistics are given in Table 6.11. (See section 6.2 for a description of the data in the table). From this table, it can be seen for instance that it can be concluded that FWNN-EXP is superior to EWNN, as the test statistic is 1.963. Similarly, it can be concluded that FWNN-80:20 is not superior to FWNN-EXP, as the test statistic is 1.700.

The first result of interest is that all techniques showed a significantly higher prediction accuracy than EWNN. This is encouraging from the point of view of determining the applicability of the other indexing techniques. If EWNN had been shown to not be significantly different from other techniques in terms of prediction accuracy, it would have indicated that no fields in cases hold knowledge more significant than other fields. As the results show however, the premise on which the techniques were chosen and implemented, namely that certain fields hold more important knowledge than others, appears to be a valid one, thus justifying the choice of the techniques.

The second point to view is that the prediction accuracy of FWNN-80:20 cannot be concluded to be significantly higher than FWNN-EXP. In Chapter 5, it was discussed that the techniques

represent a move from single global indexing to single case-specific indexing to multiple case-specific indexing. Both FWNN-80:20 and FWNN-EXP are single global indexes (but unlike EWNN, they assign more importance to certain fields.) It would be expected that they behave in much the same manner. The results achieved confirm this conclusion.

The third comparison to make is between the two case-specific techniques, VWNN and SFVW. In Chapter 5, it was described how cases have a number of solutions rather than a single one, and that SFVW retrieves for each solution separately. Examination of the cases indicated that reasons for different solutions varied within a single case, thus it could be expected that SFVW would perform better than VWNN. As table 6.11 shows, it can be concluded that SFVW does not have a significantly higher prediction accuracy than VWNN.

One possible reason for this result is simply that SFVW is not a better technique than VWNN for this application domain. However, there are other issues, which relate to the idea of validated retrieval (as described in Simoudis & Miller, 1990). Firstly, as can be seen from the matchings in Tables 6.5 & 6.6, matchings of the Top 3 cases in SFVW tend to be high. Often, many cases which taught exactly the same lesson were retrieved. In addition, two cases which each taught a different part of the solution would both have the same matching (eg 100). As a result, the top 3 cases chosen could often all teach the same lesson, but not the complete one. Thus, in situations like this, only a partial solution would have been achieved, even though the retrieval phase had found all the correct cases. In the case of VWNN, this effect was not as great, as 10 cases were used. With SFVW however, using 3 cases left more room for such mistakes.

One solution could obviously be to use 10 cases in SFVW, but this would have led to a human expert having to examine 60 (10 cases multiplied by 6 solution fields) cases to solve a single problem. A better solution would be to implement some sort of validated retrieval mechanism. For instance, cases could be grouped so that those which indicated the same solution for the same reason(s), and then only one candidate case from each of these grouping be presented. This could be considered a legitimate improvement to the SFVW technique.

Secondly, the case base was found to be inconsistent. For instance, one case might indicate that a TREM card of 3.15 should be used, while another might indicate 3.27. While this is a data consistency error, and not an indexing one, it is felt that this could have led to further degradation in performance of SFVW. For instance, take an input case (for TREM) where the actual solution was 3.27, and there were two cases in case memory which taught that the solution should be 3.15, six that taught 3.27, and all of these stored cases had the same reasons for the choice. (This is an example of the type of data inconsistency.) VWNN would most likely retrieve and choose all eight of these cases. The user would then choose the solution of 6.27, as there was a higher occurrence of it, thus coming up with the correct answer. With SFVW, it is quite possible that both 3.15 solutions would be retrieved, and only one 3.27. In this case, the solution of 3.15 would be chosen. Logically, from the data in case memory, this is not incorrect. However, an incorrect solution is still produced.

From this, it can be seen that a data inconsistency error could have led to a degradation in performance of SFVW. However, if data inconsistency was removed, this error too would be removed.

Despite the possible reasons for SFVW not out-performing VWNN, it must be concluded based on the results obtained that SFVW as it stands cannot be considered superior to VWNN. Having concluded this, the last comparison to make was between the case-specific indexing techniques (SFVW and VWNN) and the global techniques (FWNN-EXP and FWNN-80:20). Having determined in Chapter 5 that the important knowledge is not located in the same fields in each case, it was expected that case-specific techniques would out-perform those using a global technique. Examining Table 6.11 shows mixed results. Table 6.11 shows that for the conventional global weighting mechanism (FWNN-EXP), both case-specific techniques can be seen to be out-performing it. This in isolation would allow us to conclude that case-specific indexing outperforms global indexing for the chosen domain.

FWNN-80:20, however, shows a different result. From Table 6.11, it cannot be concluded that either of the case-specific indexing techniques out-perform FWNN-80:20. Considering the fact

that FWNN-EXP and FWNN-80:20 are very similar in concept, this result is surprising. However, it is felt that there is a valid reason for it. In Chapter 5, it was pointed out that FWNN-80:20 was chosen based on expert opinion and the specific case base of 90 cases used. VWNN on the other hand was chosen based only on expert opinion, without reference to the specific case base used. As such, FWNN-80:20 can be viewed as a biassed technique, as it is tailored specifically to the case base used, and not to a case base of any size or diversity. It would be expected that this bias would result in degradation of performance of FWNN-80:20 if the case base was increased in size and diversity, while the other techniques, which were chosen without bias towards the specific case base in use would not experience degradation.

It is thus felt that, in spite of the results of FWNN-80:20 (which are biassed), it can be concluded that the case-specific indexes do indicate better performance than global indexing, as is illustrated by the significantly higher performance of SFVW and VWNN over VWNN-EXP, the conventional global indexing technique.

As a conclusion in terms of comparison of the prediction accuracies, it was decided that based on prediction accuracies, the case-specific prediction accuracies would be used to compare results to other reasoners' solution performance.

Comparisons to other reasoners show that the techniques (specifically SFVW) performed fairly well :

- CBA (Gonzalez & Laureano-Ortiz, 1992) achieved an average difference percentage of 9%.
- Buta's (1994) techniques achieved accuracies of 90.4% and 84.4% on average.
- Aha & Bankert's (1994) classification achieved a highest accuracy of 88.0%.
- Battle Planner achieved 81.3% accuracy in predicting victors, 90.3% at generating either exact hots or near-misses.
- CLAVIER retrieved the "right" case 30% of the time at deployment, rising to 90% at time of publishing the paper (Hennessy & Hinkle, 1992).

- CASCADE claims 100% precision for its validated retrieval. It appears that this, however, is precision of case retrieval, not precision of solution built.
- Petrak *et al.* (1994) used error rates (the converse of accuracies, eg. accuracy of 90% (0.9) means an error rate of 10% (0.1)). They achieved error rates ranging from 0.37 to 0.83.

If one considers the above results from other reasoners, it can be said that HACA's case-specific techniques with prediction accuracies of 0.87 (87%) and 0.89 (89%) (as well as the global ones) performed well in comparison to other systems implemented. If we take the performance of the systems listed here as being representative, it can be said that the accuracy figures are satisfactory. Thus, from a CBR point of view, it can be concluded that the case-specific techniques are effective for indexing the case base.

In terms of the implications of these results for the building of an intelligent decision support system based on CBR, it can be concluded that the results indicate success. The high accuracy figures indicate that the correct information could be retrieved from case memory for a particular problem. It was shown in Chapter 4 how retrieval of relevant cases from memory could enable the building of a successful intelligent decision support system. By achieving the goal of successfully retrieving the relevant cases from case memory, success in indicating the applicability of CBR to the building of an intelligent decision support system for pre-transportation decision making for hazardous waste handling at Waste-tech (Pty) Ltd was achieved.

6.3.2.2 Case Matching and Number of Cases Retrieved

Matching figures in Table 6.2 and 6.3 show that matching increased from EWNN to FWNN-EXP to FWNN-80:20 to VWNN. As discussed in section 6.1.3, the question is whether this indicates a greater certainty in the usefulness of cases retrieved.

From the prediction accuracies (Table 6.9 and section 6.3.2.1) it can be seen that for the increase in matching figures, there was a corresponding increase in accuracy (although not always a significant increase). While for FWNN-EXP and FWNN-80:20, the pattern is not clear, the

number of cases retrieved (Table 6.4) indicates clearly for VWNN that while matching figures and prediction accuracy are higher, the number of cases retrieved does not rise correspondingly (it is actually slightly lower). This leads to the conclusion that for VWNN, the higher matching figures are indeed due to a higher certainty in the usefulness of cases retrieved, rather than simply differing matching methods.

While the number of cases retrieved does increase from EWNN to the fixed weighting techniques (FWNN-EXP and FWNN-80:20), it is still felt that with the increase in prediction accuracies, there is indication that higher matching figures are related to a greater certainty in the usefulness of retrieved cases.

For SFVW, a slightly different situation existed. Specifically, as retrieval is done on a by-field rather than by-case basis, a matching figure for whole cases could not be calculated, rather only for parts of cases. These results are shown in Table 6.5 and Table 6.6. Obviously, these cannot be compared to those in Table 6.2 and Table 6.3. Nevertheless, there are two issues to point out.

The first is that average matching for the top 3 cases (Table 6.6) was higher than for all cases retrieved above threshold (Table 6.5). This would indicate that 3 cases is a useful cutoff in terms of case selection for decision making (i.e. presentation to the user).

The second point to explain is the low matching figures for the Site and Clothing fields, as well as low retrieval for Site, Warning and Clothing fields (Table 6.7). This was not due to there being no solutions present for these fields in case memory. Rather, it was due to the way certain fields are handled in matching. Specifically, in certain situations, the content of a case does not warrant any special action. In such a situation, a default or STANDARD action is taken. An example would be "Gloves, Boots & Goggles" for protective clothing. In such a situation, STANDARD is entered in the reason field, in this case the Warnrn field. Obviously, STANDARD will not match any REPRESENTATION fields contents, thus for this field, a matching of zero will occur, and hence cases are not retrieved. However, what is specifically done here is that the matching is ignored completely, and a default action defined for solution construction. This default action

is that if no suitable solution is found in the retrieved cases, the STANDARD solution will always be adopted.

As matching figures, while comparable within an application due to the common data used, tend to be highly application specific, they will not be compared to other reasoners. However, two points can be made with regards to actual system implementation. Firstly, the higher figures for VWNN make it a more favourable technique for implementation than the three techniques preceding it. This is simply because a higher threshold for matching could be used for matching. From a user perspective this might be important, as a user might tend to place little importance on low matching figures, which might lead to distrust of the reasoner. From this perspective, we could view VWNN as the best of the four techniques shown in Table 6.2. Similarly, the high matching figures found in SFVW indicates that it would also be suitable.

Number of cases retrieved can be compared to other reasoners, and here room for possible improvement is found. While the number of cases retrieved was restricted by using a threshold for retrieval of 30%, and then restricted reasoning further by only selecting the top 10/top 3 cases, these were essentially artificial limits, chosen for ease of use rather than through some knowledge-guided technique. In addition, these retrieval figures are still fairly high when compared to those obtained by Simoudis & Miller (1990), where selectivity ranged between 1.5% and 3%. A good concept to solve this problem might be to use domain specific knowledge to validate retrieved cases (as in Validated Retrieval - Simoudis & Miller, 1990), and trim the selected case set. Once again, however, this was outside the scope of the research. As identifying the relevant cases was the only concern, no investigation was made to actively prune the retrieved case set. Nevertheless, this would be an worthwhile and valid avenue to explore.

From a system implementation point of view, this need is highlighted. In the testing with a case base of only 90 cases, even high retrieval numbers still gave a relatively small number of cases for consideration, simply due to the small case base size. In addition, in a case base of this size, a fair amount of variability was likely to be found. As such, it is unlikely that many identical cases would be found. However, if a large case base (eg. a thousand cases) was used, many identical

cases might be retrieved. A single unique case thus might be obscured by many others being retrieved. This area would definitely require further research.

6.3.2.3 Retrieval Time

Retrieval time (Table 6.1) indicates three points. Firstly, and unsurprisingly, there was negligible difference in retrieval time between the five HACA techniques. As already discussed, in terms of search, all five use a nearest neighbour approach, scanning all cases in the case base individually. It would therefore be expected that all five would have approximately the same retrieval time. The value of this result is that it indicates that none of the five indexing techniques presented any more overhead from a retrieval time point of view. As such, when evaluating the indexing techniques effectiveness in relation to each other, retrieval time is not an issue.

Secondly, in relation to actual system building, with a retrieval time of 3.2 seconds for a case base of 90 cases on an older PC, simple nearest neighbour retrieval might well be a viable technique from a retrieval time point of view. Extrapolating linearly from the results, it would be expected that a case base of 1000 cases would still only have a retrieval time of approximately 34 seconds on comparable hardware, while faster hardware could achieve significant speed-up. Thus, search of the entire case base as has been used might well be a viable method for real-world implementation.

However, compared to systems such as GRAND (Oosthuizen, 1994), where retrieval is of the order of one second, or Battle Planner, where retrieval on a 605 case base took less than a second, the implemented search technique clearly is slower. As discussed in Chapter 3, this is due to a rise in retrieval time in nearest neighbour systems corresponding to an increase in case base size. Adopting a technique which shared indexes amongst cases such as GRAND does could achieve this speed up. However, as discussed, retrieval time was not a focus of the research, thus while this indicates an area of potential future research, it does not indicate a fault in the techniques implemented.

6.3.3 Summary of Findings of the Research and Areas for Future Development

In the discussion of the results, a number of conclusions regarding the research were made, along with some areas of possible further research. These are summarised here, along with a short discussion on issues of relevance to the building of a real-world implementation.

6.3.3.1 Prediction Accuracy

In terms of prediction accuracy, the main means of measuring the techniques' effectiveness, results were as expected. The case-specific indexing techniques, VWNN and SFVW, were found to be the most effective. While SFVW yielded higher prediction accuracy than VWNN, this was not a statistically higher figure. This lack of further improvement was ascribed to inconsistency of information in the cases, and the need to refine the selection of cases, possibly through some means of validated retrieval.

A highest average prediction accuracy of 0.89 (89%) compares well to other reasoners performances, which led to the conclusion that the case-specific indexing techniques were successful for indexing the case base.

6.3.3.2 Matching and Number of Cases Retrieved

Here the main conclusion was that a combination of increased matching, increased prediction accuracy and no increase in number of cases retrieved indicated that VWNN was retrieving cases with greater certainty than other techniques.

Compared to other reasoners, techniques tended to retrieve too many cases. As specific attention was not paid to this point, this is unsurprising. It was concluded that rather than using simply a combination of a threshold plus selection of the top n (eg. 10 or 3) cases for solution building (as the techniques do), some sort of pruning, such as that used in validated retrieval, might well be useful. However, once again this was outside the scope of the research and was not pursued.

6.3.3.3 Retrieval Time

Here three main points were highlighted. The first was that as all the techniques exhibited similar retrieval time, no indexing method could be viewed as being more detrimental to retrieval speed than another.

The second point was that, as retrieval was not very slow, nearest-neighbour retrieval could well be sufficient for a real-world application.

The last point was that while being satisfactory, retrieval did not approach the speeds of the rapid searches of systems such as GRAND (Oosthuizen, 1994) or Battle Planner (Goodman, 1989). As such, the possibility of speeding up retrieval could also be an area of future research.

6.3.3.4 Issues relating to Implementing an Intelligent Decision Support System for Pre-Transportation Decision Making for Hazardous Waste Handling Using CBR

The suggested improvements given here were discovered as a result of the experiments conducted. As the implementation of such a decision support system was not the focus of the research, this discussion is included more for interest, and errors cannot be viewed as faults of the research, as they fall outside the scope of the research. In addition, they were only discovered as a result of performing the experiments, thus their discovery can be viewed as another success of the techniques designed and implemented.

6.3.3.5.1 Case Coverage

It was found, totally unsurprising, that some errors came about due to the fact that the case base did not contain the data needed to solve the current problem. Obviously, using only 90 of potentially thousands of cases, there was no hope of achieving complete coverage. Thus an immediate area of improvement in the system would be to add more cases to the case base. This is a fairly intuitive improvement, and one which most case-based reasoners would require.

However, similar case base sizes were used elsewhere, eg. CARMA, CLAVIER, ARCHIE and CASCADE, and this case base size was sufficient to test the techniques under consideration.

6.3.3.5.2 Inconsistency of Case Base

It was found during testing that while no serious errors occurred, there was some inconsistency in terms of solution provision, i.e. given the same values in REPRESENTATION fields, occasionally, different actions were taken. This could probably be attributed to different cases being handled by different consultants. This inconsistency raises two points.

Firstly, if the system were to be implemented in the real world, these inconsistencies would have to be removed, presumably by having one expert make changes whenever inconsistencies were encountered and, possibly, discarding certain incorrect or "less correct" solutions.

Secondly, the system has in fact already been useful as it has highlighted these inconsistencies. This already points to the usefulness of such a system, as it would help experts to validate their solutions in light of other expert opinion.

6.3.3.5.3 Incomplete Differentiation of Solutions

An interesting point was found in solution construction. This was that, as yet, certain expert knowledge is still not contained in the case despite the NEW fields being added. If we look at Table 6.8, it can be seen that compared to other solutions, the HAZCHEM and TREM fields accuracy was fairly low. This is due to the fact that different solutions (eg different TREM card numbers) were provided in different cases, yet reasons given for them (i.e. lists of REPRESENTATION fields) were the same or similar. What is happening is that the differentiation between these solutions was not being completely defined by the contents of the NEW fields.

Whether this problem was due to inconsistency in application of knowledge by experts or simply insufficient knowledge being explicitly listed in the cases is unclear. However, it is suspected that

insufficient knowledge is currently contained in the cases to completely differentiate between all possible solution reasons.

It can thus be said that a further refinement of the case base would be necessary, either through addition of fields, or through generalisation which will be discussed next.

6.3.3.5.4 Generalisation

In examining certain explicit requirements for solutions (eg. requirements of TREM cards) as set out in guidelines at Waste-tech, it can clearly be seen that while cases work at a very specific level (eg. contains Sulphuric Acid) many solutions actually act at a more general level (eg. contains strong acid). Having ascertained that the indexing strategy works at a case specific level, this knowledge of generalisation indicates that adding generalisation to the system, much the same way that CHEF (Hammond, 1989) generalises, could well greatly enhance performance of the system.

However, especially in this area, extensive testing would have to be performed. As shown in CHEF, when generalisations are used, errors occur easily (the BEEF-AND-BROCCOLI episode). In a domain such as CHEF, such an error can be treated as a learning experience. However, in this domain, there is far more risk involved, and an incorrect generalisation could have disastrous ramifications. As such, while generalisation might appear desirable, exceptionally thorough evaluation would have to occur before such a technique was viable.

6.3.3.5.5 Extraction of Common Indexes

A final point links back to retrieval time. In GRAND, we see how common features are shared to improve retrieval speed by decreasing the amount of matching. It is proposed that in the domain under study, the same broad concept could be implemented to improve speed. Specifically, we have seen how in SFVW, each reason field is used as a separate index (i.e. the case is indexed six times). Effectively what happens is that only those fields of each stored case are examined. The slowdown in the technique occurs due to the fact that each case has to be examined on these fields.

It is proposed that to speed up performance, these indexes would simply have to be shared. This would be done by finding all cases with a common index, storing this index once, and then having it point to all the cases sharing that index. At retrieval time, the input case would only have to examine this index once, and then automatically have a matching for the field referred to by the index for 50 cases, instead of examining each case individually. For example, if 50 cases had the disposalrn field containing the values :

Flammability : Positive.

Flash-Point : Positive.

this index could be stored once and made to point to all 50 cases, thus greatly improving retrieval time. Of course, there are many additional complications, such as ensuring that all these cases then actually contain the same disposal instruction. Nevertheless, the proposal appears viable.

CONCLUSION

The aim of this research was to investigate and apply indexing in case-based reasoning. In conjunction, and to enable research on indexing, case representation was also researched so as to facilitate the design and implementation of the indexing techniques. A domain where application of CBR was applicable was chosen to allow testing of the indexing techniques. This domain was pre-transportation decision making for hazardous waste handling. A case representation and a number of indexing techniques were then implemented which were applicable to the domain.

In order to place research on indexing in context with CBR research, an overview of the CBR field was conducted. Due to the fact that case representation must be successfully implemented before indexing can be tested, case representation was also researched in some depth with respect to the practical area under consideration. Research areas such as case retrieval and adaptation were not examined. The application domain was chosen in order to allow for the implementation and testing of indexing techniques, and due to it's suitability for the application of CBR.

As such, a literature survey was performed to examine :

- A general overview of CBR.
- The theory and techniques of Case Representation in detail.
- The theory and techniques of Indexing in detail.

In addition, the decision making process at Waste-tech (Pty) Ltd, the chosen domain for application of the techniques under research, was examined in order to assess how an intelligent decision support could be incorporated into the current decision making process.

Following this, five indexing techniques were developed, these being :

- Equal Weight Nearest Neighbour (EWNN).
- Fixed Weight Nearest Neighbour using Expert Judgement (FWNN-EXP).
- Fixed Weight Nearest Neighbour using the 80-20 Rule (FWNN-80:20).
- Variable Weight Nearest Neighbour (VWNN).

- Separate Field Variable Weight (SFVW).

These represent an evolution from a single, global indexing technique to a single, case-specific indexing technique to multiple, case-specific indexing of cases. The Fixed Weight Nearest Neighbour using the 80-20 Rule and Separate Field Variable Weight techniques were developed originally by the author and were not reported previously to the best of his knowledge in the CBR literature. Of the five techniques, Equal Weight Nearest Neighbour and Fixed Weight Nearest Neighbour using the 80:20 Rule were implemented in a commercial case-based reasoning shell, ESTEEM, while all five techniques were implemented as custom modules in a system called HACA (The Hazardous Chemicals Advisor) which was constructed using Delphi 1.0, a Windows programming language.

For testing, case base size and tests used were determined after examining the testing procedures found in other reasoners published in the literature. As case adaptation was not a research focus, solution construction was undertaken by hand, thus the reasoner followed a decision-support role as suggested in Kolodner (1991). Testing was undertaken with four goals in mind :

- HACA's implementation of techniques was compared with ESTEEM's to validate the custom implementation's behaviour.
- The five indexing techniques designed and implemented in HACA were compared to assess which performed best as an indexing technique.
- The performance of indexing techniques designed (with particular attention to the one considered best) was compared to the performance of other reasoners to assess whether any of the techniques designed could be considered satisfactory in the context of other implementations of case-based reasoning.
- The implications which the results have for the actual building of an intelligent decision support system in the domain using case-based reasoning were assessed.

Using modified Waste-tech data sheets as cases, a case base of ninety cases was constructed. Using ten cases from the case base as input cases, tests were then run on the five techniques, and the following data were captured for analysis :

- Time taken for case retrieval.
- Average matching of cases retrieved. (For Separate Field Variable Weight, this was calculated for each solution field).
- Average matching of the top ten (or less) cases retrieved. (For Separate Field Variable Weight, this was the top three (or less), calculated again for each solution field).
- The number of cases retrieved above the threshold. (For Separate Field Variable Weight, this was calculated again for each solution field). The threshold was set to 0.3 (30%).
- The prediction accuracy for each case. For Separate Field Variable Weight, prediction accuracy was calculated for each case as a whole, as well as for each solution field.

As the focus of the research was on indexing, prediction accuracy was used as the primary means of assessing and comparing performance. As such, prediction accuracies were compared using paired *t*-tests, as described in Pollard (1977).

In terms of validation of HACA's implementation of techniques, the results showed that HACA's implementation performance was not significantly different to that of ESTEEM in terms of accuracy, but was faster. This indicates that HACA's implementation could be considered to be behaving correctly.

In terms of accuracy of solutions produced, it was concluded that case-specific indexing techniques (VWNN and SFVW) were the most effective, and that performance was possibly degraded due to data inconsistencies.

Comparison to results of other reasoners showed that the case-specific techniques produced comparable solution accuracy. This led to the additional conclusion that in comparison to other reasoners, the techniques could also be considered satisfactory. As such, it can be concluded that the case-specific techniques were effective as indexing techniques.

Retrieval time results showed that all techniques performed roughly equally in comparison to each other. In the context of the research (indexing of cases), this was considered a favourable result

as it indicated that none of the indexing techniques implemented had an adverse effect on retrieval time in comparison to the other techniques. It was noted that research focussed on retrieval time could well lead to faster retrieval times but, as this was not within the scope of the research, it was not considered a drawback in the results.

In terms of matching, it was found that matching figures increased from EWNN to FWNN-EXP to FWNN-80:20 to VWNN (SFVW could not be compared). By examining prediction accuracy and number of cases retrieved, it was concluded that for VWNN, this increase in matching could be considered to be an increased certainty in the usefulness of the retrieved cases. This lent further credence to the conclusion that VWNN was a more satisfactory indexing technique than EWNN, FWNN-EXP or FWNN-80:20.

For number of cases retrieved, a potential improvement was identified. This is that compared to some other reasoners which concentrated on pruning the set of retrieved cases, selectivity could be more effective, resulting in less cases being retrieved for presentation to the decision maker. However, as with retrieval time, this was outside the scope of this research.

A number of future improvements (other than those mentioned) were found. Unsurprisingly, a case base of ninety cases was found to give incomplete coverage, but nevertheless to be sufficient for experimental testing purposes, as indicated by the literature. The data in cases were found in some cases to be inconsistent, while it was also found that certain expert knowledge is not contained in the cases. It was also noted that generalisation of information might be viable, and that common indexes could be used to speed up scanning of case memory.

However, the listing of these possible improvements should not be viewed as errors in the research. Examination shows that they fall into two areas :

- Problems which would be solved when an actual system was implemented (eg. increase case base size, inconsistency of case base).
- Issues beyond the scope of the research (eg. retrieval time, number of cases presented, generalisation, common indexes).

It can be concluded from this that the improvements suggested fell outside the scope of the research, and are areas for future research.

The purpose of the research was to investigate and implement indexing techniques for CBR. From the discussion and results presented in the dissertation, it has been shown that indexing techniques were designed based on established CBR principles and implemented in the suitable and significant from and application point of view domain of pre-transportation decision making for hazardous waste handling. Results indicate that the performance of the techniques matches the theoretical basis for their design, and is satisfactory in comparison to other CBR research.

REFERENCES

Aamodt, A. 1993. Explanation-Driven Retrieval, Reuse and Learning of Cases. In *First European Workshop on Case-Based Reasoning (EWCBR-93) Presentations and Posters*, University of Kaiserslautern, Germany. Eds Richter, M. M., Wess, S., Althoff, K-D., Maurer, F. November 1-5 1993: 279-284.

Aamodt, A., Plaza, E. 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. AI Communications, 7(1):39-52. Downloaded from http://www.mines.u-nancy.fr/~gueniffe/CoursEMN/I31/AICom/AICom.html

Aha, D. W., Bankert, R. L. 1994. Feature Selection for Case-Based Classification of Cloud Types: an Empirical Comparison. In *Papers from the 1994 AAAI Workshop,* Seattle, Washington, AAAI Press, Menlo Park, California, Technical Report WS-94-01, August 2 1994: 106-112.

Aha, D. W., Bankert, R. L. 1997. Cloud Classification Using Error-Correcting Output Codes. *Artificial Intelligence Applications: Natural Resources, Agriculture and Environmental Science* 11:1, 13-28. Downloaded from http://www.aic.nrl.navy.mil/~aha/pub-details.html.

Allen, B. P. 1994. Case-Based Reasoning: Business Applications. *Communications of the ACM,* Vol. 37, No. 3, March 1994: 40-42.

Andersson, S. 1994. Safe Transport of Dangerous Goods: Road, Rail or Sea? A Screening of Technical and Administrative Factors. *European Journal of Operational Research,* Vol. 75. 499-507.

Barletta, R. 1991. An Introduction to Case-Based Reasoning. *AI Expert*, Vol. 6, No. 8, August 1991: 43-49.

Bayles, S. J., Das, B. K. 1994. Using Artificial Intelligence to Support Traffic Flow Management Problem Resolution. In *Papers from the 1994 AAAI Workshop,* Seattle, Washington, AAAI Press, Menlo Park, California, Technical Report WS-94-01, August 2 1994: 133-137.

Beroggi, G. E. G. 1994. A Real-Time Routing Model for Hazardous Materials. *European Journal of Operational Research*, Vol. 75: 508-520.

Branting, L. K., Hastings, J. D. 1994. An Empirical Evaluation of Model-Based Case Matching and Adaptation. In *Papers from the 1994 AAAI Workshop,* Seattle, Washington, AAAI Press, Menlo Park, California, Technical Report WS-94-01, August 2 1994: 72-78.

Buta, P. 1994. Mining for Financial Knowledge with CBR. *AI Expert*, February: 34-41.

Caruso, C., Colorni, A., Paruccini, M. 1993. The Regional Urban Solid Waste Management System: A Modelling Approach. *European Journal of Operations Research*, Vol. 70: 16-30.

Chan, C. W., Tontiwachwuthikul, P. 1995. Expert System for Solvent Selection of CO_2 Separation Processes. *Expert Systems with Applications*, Vol. 8, No. 1: 33-46.

Chaturverdi, A. R. 1993. Acquiring Expert Knowledge in a Complex Domain. *Expert Systems with Applications,* Vol. 6, No. 1: 23-35.

Chi, R. T., Chen, M., Kiang, M. Y. 1993. Generalized Case-Based Reasoning System for Portfolio Management. *Expert Systems with Applications,* Vol. 6, No. 1: 67-76.

Cohen, P. R. 1989. Evaluation and Case-Based Reasoning. In *Proceedings: Workshop on Case-Based Reasoning (DARPA), Pensacola Beach, Florida*. Ed. Hammond, K. San Mateo, CA: Morgan Kaufmann. 168-172.

Cunningham, P. Slattery, S. 1993. Knowledge Engineering Requirements in Derivational Analogy. In *First European Workshop on Case-Based Reasoning (EWCBR-93) Presentations and Posters*, University of Kaiserslautern, Germany. Eds Richter, M. M., Wess, S., Althoff, K-D., Maurer, F. November 1-5 1993: 108-113.

Department of Environment Affairs. 1992. *Hazardous Waste in South Africa.* Executive summary, Vol.1: Situation Analysis, Vol. 2: Technologies, Vol. 3: Policy, Vol 4: Legislative Options, Vol. 5: Impact Assessment. Ed. R. G. Noble. CSIR, Pretoria.

Glass, R. L. 1995. A Structure-Based Critique of Contemporary Computing Research. *Journal of Systems Software*, 28: 3-7.

Gonzalez, A. J., Laureano-Ortiz, R. 1992. A Case-Based Reasoning Approach to Real Estate Property. *Expert Systems With Applications*, Vol. 4: 229-246.

Goodman, M. 1989. CBR in Battle Planning. In *Proceedings: Workshop on Case-Based Reasoning (DARPA), Pensacola Beach, Florida*. Ed. Hammond, K. San Mateo, CA: Morgan Kaufmann. 264-269.

Gupta, U. G. 1994. How Case-Based Reasoning Solves New Problems. Interfaces 24: 6, November-December: 110-119.

Haastrup, P. 1994. Overview of Problems of Risk Management of Accidents with Dangerous Chemicals in Europe. *European Journal of Operational Research*, Vol. 75, 488-498.

Hammond, K. J. 1989. *Case-Based Planning: Viewing Planning as a Memory Task.* Ed. B. Chandrasekaran. San Diego. Academic Press.

Hansen, J. V., Meservy, R. D., Wood, L. E. 1994. Indexing and Tree Pruning Concepts to Support Case-Based Reasoning. *Omega, International Journal of Management Science*, Vol. 22, No. 4: 361-369.

Harmon, P. 1991. Case-Based Reasoning. *Intelligent Software Strategies,* Vol. VII, No. 11, November: 1-15.

Helton, T. 1991. The Hottest New AI Technology: Case-Based Reasoning. *The Spang Robinson Report on Artificial Intelligence,* Vol. 7, No. 8: 4-7.

Hennessy, D. Hinkle, D. 1992. Applying Case-Based Reasoning to Autoclave Loading. *IEEE Expert,* Vol. 7, No. 5: 21-26.

Hinrichs, T. R. 1988. Towards Architecture for Open World Problem Solving. In *Proceedings: Workshop on Case-Based Reasoning (DARPA), Clearwater, Florida.* San Mateo, CA: Morgan Kaufmann. 182-189.

Hushon, J. M. 1991. Application of Intelligent Computer Technology to Hazardous Chemical Management. *The Journal of Knowledge Engineering*, Vol. 4, No. 3: 41-47.

Jones, E. K., Roydhouse, A. 1994. Iterative Design of Case Retrieval Systems. In *Papers from the 1994 AAAI Workshop,* Seattle, Washington, AAAI Press, Menlo Park, California, Technical Report WS-94-01, August 2 1994: 150-156.

Ketler, K. 1993. Case-Based Reasoning: An Introduction. *Expert Systems with Applications,* Vol. 6, No. 1: 3-8.

Kolodner, J. L. 1987. Extending Problem Solver Capabilities Through Case-Based Inference. *In Proceedings: 4th Annual International Machine Learning Workshop.* Morgan Kaufmann. 21-30.

Kolodner, J. L. 1991. Improving Human Decision Making Through Case-Based Decision Aiding. *AI Magazine* Vol. 12, No. 2: 52-68.

Kolodner, J. L., Mark, W. 1992. Guest Editors' Introduction - Case-Based Reasoning. *IEEE Expert,* Vol. 7, No. 5: 5-6.

Kolodner, J. L. 1993. *Case-Based Reasoning.* San Mateo. Morgan Kaufmann.

Krovvidy, S., Wee, W. G., Summers, R. S, Coleman, J. J. 1991. An AI Approach for Wastewater Treatment Systems. *Journal of Applied Intelligence*, Vol. 1, No. 3: 247-261.

Langley, P., Sage, S. 1994. Oblivious Decision Trees and Abstract Cases. In *Papers from the 1994 AAAI Workshop,* Seattle, Washington, AAAI Press, Menlo Park, California, Technical Report WS-94-01, August 2 1994: 113-117.

Leake, D.B. 1996. CBR in Context: The Present and Future. In *Leake, D., editor, Case-Based Reasoning: Experiences, Lessons, and Future Directions*. AAAI Press/MIT Press, 1996. Downloaded from http://www.cs.indiana.edu/hyplan/leake/papers/INDEX.html.

Leake, D.B., Kinley, A., Wilson, D. 1997. A Case Study of Case-Based CBR.. In *Proceedings of the Second International Conference on Case-Based Reasoning,* Springer Verlag, Berlin, 1997. Downloaded from http://www.cs.indiana.edu/hyplan/leake/papers/INDEX.html.

Liang, T.-P. 1993. Guest Editors' Note - Case-Based Reasoning and Its Applications. *Expert Systems with Applications,* Vol. 6, No. 1: 1-2.

List, G. F., Mirchandani, P. B., Turnquist, M. A., Zografos, K. G. 1991. Modelling and Analysis for Hazardous Materials Transportation: Risk Analysis, Routing/ Scheduling and Facility Location. *Transportation Science*, Vol. 25, No. 2: 100-114.

List, G. F., Turnquist, M. A. 1993. Routing and Emergency Response Team Sitting for High-Level Radioactive Waste Shipments. *Paper Prepared for the 1994 Annual Meeting of the Transportation Research Board*. August 1993: 26 pp.

McCartney, R. 1993. Episodic Cases and Real-Time Performance in a Case-Based Planning System. *Expert Systems with Applications,* Vol. 6, No. 1: 9-22.

Mitchell, G. 1993. *The Practice of Operational Research*. J. Wiley.

Navinchandra, D. 1988. Case-Based Reasoning in CYCLOPS, a Design Problem Solver. In *Proceedings: Workshop on Case-Based Reasoning (DARPA), Clearwater, Florida*. San Mateo, CA: Morgan Kaufmann. 286-301.

O'Leary, D.E. 1993. Verification and Validation of Case-Based Systems. *Expert Systems with Applications,* Vol. 6, No. 1: 57-66.

Oosthuizen, G. D. 1994. A Dynamic Indexing Mechanism for Memory-Based Reasoning. In *Proceedings of the Intern. AMSE Conference "Intelligent Systems"*, Pretoria, South Africa, AMSE Press, Vol. 28-30, Sept 1994: 127-136.

Owens, C. 1988. Domain-Independent Prototype Cases for Planning. In *Proceedings: Workshop on Case-Based Reasoning (DARPA), Clearwater, Florida*. San Mateo, CA: Morgan Kaufmann. 302-311.

Pazzani, M. J. 1989. Indexing Strategies for Goal Specific Retrieval of Cases. In *Proceedings: Workshop on Case-Based Reasoning (DARPA), Pensacola Beach, Florida*. San Mateo, CA: Morgan Kaufmann. 31-35.

Pearce, M., Goel, K., Kolodner, J. L., Zimring, C., Sentosa, L., Billington, R. 1992. Case-Based Design Support - A Case Study in Architectural Design. *IEEE Expert,* Vol. 7, No. 5: 14-20.

Petrak, J., Trappl, R., Furnkranz, J. 1994. *The Possible Contribution of AI to the Avoidance of Crises and Wars: Using CBR Methods with the KOSIMO Data Base of Conflicts.* Austrian Research Institute for Artificial Intelligence.

Pollard, J. H. 1977. *A Handbook of Numerical and Statistical Techniques.* Cambridge University Press.

Radermacher, F. J, Riekert, W.-F., Page, B., Hilty, L. M. 1994. Trends in Environmental Information Processing. In *Proceedings of the IFIP Congress 94*, Vol. 2. 1994. Elsevier Science Publishers B.V. (North Holland).

Ram, A. 1989. Incremental Learning of Paradigmatic Cases. In *Proceedings: Workshop on Case-Based Reasoning (DARPA), Pensacola Beach, Florida*. San Mateo, CA: Morgan Kaufmann. 304-308.

Rauscher, H. M., Hacker, R. 1989. Overview of Artificial Intelligence Applications in Natural Resource Management. *The Journal of Knowledge Engineering*, Vol. 2, No. 3, Fall 1989: 30-42.

Repede, J. F., Bernardo, J. J. 1994. Developing and Validating a Decision Support System for Locating Emergency Medical Vechiles in Louisville, Kentucky. *European Journal of Operational Research.* Vol. 75: 567-581.

Riesbeck, C. K., Schank, R. C. 1989. *Inside Case-Based Reasoning*. Hillsdale, New Jersey. Lawrence Erlbaum Associates.

Schank, R.C. 1988. Reminding and Memory. In *Proceedings: Workshop on Case-Based Reasoning (DARPA), Clearwater, Florida*. San Mateo, CA: Morgan Kaufmann. Downloaded from http://online.loyno.edu/cisa494/papers/Schank.html.

Selfridge, M., Cuthill, B. 1989. Retrieving Relevant Out-Of Context Cases: A Dynamic Memory Approach to Case-Based Reasoning. In *Proceedings: Workshop on Case-Based Reasoning (DARPA), Pensacola Beach, Florida*. San Mateo, CA: Morgan Kaufmann. 313-317.

Shinn, H. S. 1988. Abstractional Analogy: A Model of Analogical Reasoning. In *Proceedings: Workshop on Case-Based Reasoning (DARPA), Clearwater, Florida*. San Mateo, CA: Morgan Kaufmann. 370-387.

Simoudis, E., Miller, J. S. 1990. Validated Retrieval in Case-Based Reasoning. In *Proceedings of AAAI-90*. Cambridge, MA:AAAI Press/MIT Press. 310-315.

Simoudis, E. 1992. Using Case-Based Retrieval for Customer Technical Support. *IEEE Expert,* Vol. 7, No. 5: 7-12.

Slade, S. 1991. Case-Based Reasoning: A Research Paradigm. *AI Magazine,* Vol. 12, No. 1: 42-55.

Stottler, R. H., Henke, A. L., King, J. A. 1989. Rapid Retrieval Algorithms for Case-Based Reasoning. *Eleventh International Joint Conference on Artificial Intelligence, Detroit, Michigan USA*. 20-25 August 1989: 233-237.

Sycara, K. P. 1991. Problem restructuring in negotiation. *Management Science,* Vol. 37, No. 10: 1248-1268.

Sycara, K. P. 1993. Machine learning for intelligent support of conflict resolution. *Decision Support Systems* 10: 121-136 .

Thompson, V. 1997. Corporate Memories - New Reasoning engines and intelligent agents help companies manage their enterprise-wide knowledge resources. *Byte, International Issue,* September 1997. McGraw-Hill.

Turban, E. 1991. *Decision Support and Expert Systems - Management Support Systems.* New York. MacMillan. 2nd Edition.

Wall, R. S., Donahue, D., Hill, S. 1988. The Use of Domain Semantics for Retrieval and Explanation in Case-Based Reasoning. In *Proceedings: Workshop on Case-Based Reasoning (DARPA), Clearwater, Florida.* San Mateo, CA: Morgan Kaufmann. 447-461.

Warwick, C. J., Mumford, J. D., Norton, G. A. 1993. Environmental Management Expert Systems. *Journal of Environmental Management*, Vol. 39: 251-270.

Watson, I. 1995. Case-based reasoning development tools: A Review. Available on the World Wide Web at http://www.salford.ac.uk/docs/depts/survey/staff/IWatson/cbrtools.htm

Wettschereck, D., Aha, D., Mohri, T. 1997. A Review and Empirical Evaluation of Feature Weighting Methods for a Class of Lazy Learning Algorithms. Artificial Intelligence Review 11: 273-314. Downloaded from http://www.aic.nrl.navy.mil/~aha/pub-details.html.

APPENDIX

Table 1 : Case Representation

Field Name	**Data Sheet Field**	**Case Section**	**Valid Entries**	**Description of Contents**
sheetno	Sheet Number	REPRESENTATION	A unique six-digit number	Uniquely identifies the datasheet/case.
coname	Company	REPRESENTATION	A list of characters	The company name.
constit	Known/ Expected Constituents	REPRESENTATION	A list of known or suspected constituents prior to analysis	The number of constituents varies per case. The constituents are all stored in one field which is parsed on retrieval to construct an array of constituents.
wstate	Description of Waste	REPRESENTATION	One of five values - SOLID - SOLID/SLUDGE - SLUDGE - SLUDGE/LIQUID - LIQUID	
odour	Odours	REPRESENTATION	One of five values - STRONG - MEDIUM - WEAK - FAINT - NONE	

ph	ph	REPRESENTATION	A value from 0 - 14 or NO TEST	A value of NO TEST is given to all laboratory tests which are not performed. Fields are not left blank.
acid	% Acid equivalent	REPRESENTATION	ratio or NO TEST	
lime	Lime equivalent	REPRESENTATION	ratio or NO TEST	
flamm	Flammability	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST	
flash	Flash-point	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST	
ash1	Ash blend 1:1	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST	
ash2	Ash blend 2:1	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST	
ashn	Ash blend :1	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST	

vols	% Volatiles	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST or a number	
solids	% Solids	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST or a number	
other		REPRESENTATION	A test name	
arsenicsp	Arsenic spot	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST or a number	
cyanidesp	Cyanide spot	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST or a number	
hvymetalsp	Heavy metals spot	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST	
chromesp	Chrome spot	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST or a number	
sulphidesp	Sulphide spot	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST	

fluoridesp	Fluoride spot	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST	
chloridesp	Chloride spot	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST or a colour	
nitratesp	Nitrate spot	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST	
phosphatsp	Phosphate spot	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST	
sulphatesp	Sulphate spot	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST	
ammoniasp	Ammonia spot	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST	
cyanide	Cyanide	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST or a number	

arsenic	Arsenic	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST or a number	
lead	Lead	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST or a number	
cadium	Cadium	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST or a number	
mercury	Mercury	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST or a number	
copper	Copper	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST or a number	
zinc	Zinc	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST or a number	

anions	Anions	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST or a number	
cations	Cations	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST or a number	
phenols	Phenols	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST or a number	
volorgs	Volatile organics	REPRESENTATION	One of three values - POSITIVE - NEGATIVE - NO TEST or a number	
disposal	D i s p o s a l instructions	SOLUTION	The list of disposal instructions	As with constituents, there is a variable number of disposal instructions per case. These are stored in a single field which is parsed on retrieval.

disposalrn	NEW	SOLUTION	A list of reasons for the disposal instructions	There is one or more reasons for each disposal instruction eg. a pH of 0 means add lime. This field stores all the reasons for the disposal instructions, where each reason is a field name and value (eg. pH 0). The field is parsed on retrieval.
site	Disposal site	SOLUTION	The disposal site	
sitern	NEW	SOLUTION	A list of reasons	The same structure as disposalrn.
hazchem	Hazchem decal no.	SOLUTION	The hazchem decal number	The Hazchem decal on the vehicle identifies the type of. waste
hazchemrn	NEW	SOLUTION	A list of reasons	
trem	TREM Card	SOLUTION	The TREM card number	This is a procedure sheet the driver carries identifying the type of waste carried and the steps to be taken in the event of emergency.
tremrn	NEW	SOLUTION	A list of reasons	
warn	Hazard Warning	SOLUTION	Warnings for handling of waste	
warnrn	NEW	SOLUTION	A list of reasons	
clothing	Protective Clothing	SOLUTION	A list of protective clothing	
clothingrn	NEW	SOLUTION	A list of reasons	

HAZARDOUS WASTE ANALYTICAL DATA SHEET

000153

Lab. No. .....

Company ..... Date .....

Address ..... Tel. .....

..... Fax .....

..... Contact .....

Postal Address .....

Expected volumes ..... Frequency .....

Process generating waste .....

.....

.....

Known/expected constituents .....

.....

.....

Classification of waste			
Description of waste	SOLID	SLUDGE	LIQUID
Odours	STRONG	WEAK	NONE

Laboratory analysis (tick which required)

☐ ph	☐ Arsenic spot	☐ Cyanide
☐ % Acid equivalent	☐ Cyanide spot	☐ Arsenic
☐ Lime equivalent	☐ Heavy metals spot	☐ Lead
☐ Flammability	☐ Chrome spot	☐ Cadium
☐ Flash-point	☐ Sulphide spot	☐ Mercury
☐ Ash blend 1:1	☐ Fluoride spot	☐ Copper
☐ Ash blend 2:1	☐ Chloride spot	☐ Zinc
☐ Ash blend :1	☐ Nitrate spot	☐ Anions
☐ % Volatiles	☐ Phosphate spot	☐ Cations
☐ % Solids	☐ Sulphate spot	☐ Phenols
☐	☐ Ammonia spot	☐ Volatile organics

Comments .....

.....

.....

.....

Disposal instructions .....

.....

.....

Disposal site .....

Hazchem decal no. ..... TREM Card .....

Hazard warning .....

Protective clothing .....

Consultant ..... Depot ..... Date .....

Analyst ..... Date ..... Chemist ..... Date .....

Hazardous Waste Manager ..... Date .....

Figure 1 : Waste-tech data sheet