An

AI *Perspectives*

Report

Series Editor: Alex Goodall

A Review of Industrial Case-Based Reasoning Tools

by

Klaus-Dieter Althoff, Eric Auriol, Ralph Barletta & Michel Manago

AI *Intelligence* PO Box 95 Oxford OX2 7XL United Kingdom
Tel: +44 1865-791 600 Fax: +44 1865-791 007 Email: aii@aiintelligence.com

ISBN 1 898804 01 X

An

AI *Perspectives*

Report

Series Editor: Alex Goodall

A Review of Industrial Case-Based Reasoning Tools

by

Klaus-Dieter Althoff, Eric Auriol, Ralph Barletta & Michel Manago

Published by

AI *Intelligence*

A Reporting, Analysis and Monitoring Service for AI and KBS

About the Authors

Dr Klaus-Dieter Althoff has published a number of research papers on learning expert systems, CBR and integrated architectures. He has participated in numerous projects on integrated architectures and CBR at the University of Kaiserslautern and a small German company. He has been involved in a number of German and international events on CBR, Knowledge Acquisition and Machine Learning as co-chair/programme committee member/co-organiser. He is one of the founders of the European Workshop on CBR (EWCBR). Currently, he is project leader of the INRECA Esprit project for the University of Kaiserslautern.

✉: *althoff@informatik.uni-kl.de* ☎: *+49 631-205 3360* 🖷: *+49 631-205 3357*

Eric Auriol worked at Thomson until 1992 after which he spent 18 months at the University of Kaiserslautern in Professor Michael Richter's research group. He is currently completing his PhD on integration of induction and CBR at the French *Institut National de Recherche en Informatique et Automatique* (INRIA) under the supervision of Professor Edwin Diday and at Acknosoft. Author of 10 publications on CBR and service support systems, he has been actively involved in developing and installing CBR applications in engineering (service support for plastic injections press robots, maintenance of Boeing 737 aircraft engines).

✉: *acknowledge@applelink.apple.com*

Ralph Barletta is currently President of Case Data Solutions, Inc., a US-based software consulting company specialising in building CBR, database mining and client-server computing systems. He has been doing research and applications development in CBR and machine learning for the past 7 years and was in charge of the development of the ReMind CBR tool. Ralph received his M.S. in Computer Science from Rutgers University in 1988, and has since published over a dozen papers on CBR and machine learning.

✉: *71664.1210@compuserve.com* ☎: *+1 508-443 7594* 🖷: *+1 508-443 7594*

Michel Manago has been developing induction and CBR technology for over 10 years. After graduating (1983) from the University of Illinois, he obtained his PhD (Integration of Numeric & Symbolic Machine Learning Techniques) in 1988 from Paris University and carried out post-graduate research at Stanford. He founded Acknosoft in 1991. Author of over 30 related publications, Michel is co-chairman of a number of international workshops on CBR. In 1992 he received the prestigious *Challenge* + award from HEC (France's top business school) and received first prize for innovative software applications at the German National AI conference in 1995.

✉: *acknowledge@applelink.apple.com* ☎: *+33 1-44 24 88 00* 🖷: *+33 1-44 24 88 66*

About the Publisher

AI *Intelligence* is a reporting, analysis and monitoring service covering industrial and commercial aspects of AI and knowledge-based systems. It publishes reports - the AI *Perspectives* series - and produces a monthly newsletter, AI *Watch*, that reports on and analyses current events in the field.

To be kept informed of future reports or to suggest topics of particular interest to you, please contact the Series Editor, Alex Goodall. We would also welcome any comments on this report.

AI *Intelligence*
PO Box 95
OXFORD OX2 7XL
United Kingdom

☎: +44 1865-791 600
🖷: +44 1865-791 007
✉: alex@aiintelligence.com

An

AI *Perspectives*

Report

Series Editor: Alex Goodall

A Review of Industrial Case-Based Reasoning Tools

by

Klaus-Dieter Althoff, Eric Auriol, Ralph Barletta & Michel Manago

This Report describes the background and workings of CBR technology and compares it to other, related technologies. The main body of the Report consists of a detailed evaluation of five commercial CBR tools based on twenty technical criteria and eight ergonomic ones. To help assess the tools against these criteria, eleven tests were applied based on case data from four different domains. Many other subjective tests were also applied.

Published by

AI *Intelligence*

PO Box 95
OXFORD
OX2 7XL
United Kingdom

Tel: +44 1865-791 600
Fax: +44 1865-791 007
Email: aii@aiintelligence.com

First published 1995

ISBN 1 898804 01 X

Preface

The idea is simple and obvious.

The implications are far-reaching and exciting.

Case-Based Reasoning (CBR) is one of those rare technologies whose principles can be explained in a single sentence: "To solve a problem, remember a similar problem you have solved in the past and adapt the old solution to solve the new problem".

Try explaining the principles of neural networks, genetic algorithms or even expert systems in under twenty five words: I do not think it possible.

Another positive feature of CBR is that the idea is obvious - once it has been pointed out, of course. We all solve problems using past experience, so why shouldn't a computer?

So the CBR marketing people have a pretty easy time of it: the CBR concept is appealing, obvious, and easily explained; there are many good case studies to refer to; and the tools are generally robust and effective for use in building industrial-strength applications.

But the "technology evaluator" people have a rather harder time. They need to go beyond the simple, surface ideas and understand just ***how*** *a "similar" past problem can be remembered, just what is meant by "adapt the old solution to the new problem", and even, how do you go about* ***storing*** *past problems (and what are they, anyway?). And then, if they can get to grips with all that, the technology evaluators need to know about the commercialisation of the technology. What tools are available? How do they differ from each other? How should you go about comparing them? How does CBR compare with other, similar technologies?*

Given the context of this Preface it should be pretty clear what comes next.....

This Report has been written to answer ***exactly*** *those question, and ignoring the obvious accusations of bias, I have no hesitation in expressing my opinion that the authors and the evaluation team have done a truly first-rate job on all fronts. The explanations of the technology make the underlying complexities quite apparent, yet convey the right amount of technical detail; and the evaluations of the tools have been comprehensive and thorough.*

Despite the many applications of CBR listed in the ***Executive Summary****, this is still a very young technology whose potential is barely being touched. Although the Report does not oversell the technology in any way (on the contrary, it often downplays it by pointing out the problems that researchers are still working on), I still came away with an understanding of its capabilities that makes me very optimistic. I feel we could well see some quite remarkable developments and exiting applications over the next few years.*

But you don't have to take my word for it. You now have access to all the information to be able to form your own opinion. This is our aim in publishing **AI** *Perspectives reports.*

Alex Goodall (Series Editor for **AI** *Perspectives*)

Oxford, February 1995

Erratum: Correct Table of Contents is as follows:

TABLE OF CONTENTS

About this Document

Ever since the term **Case-Based Reasoning** (CBR) was coined at the beginning of the 'eighties, there has been a steady growth of interest in the topic. At the beginning of the 'nineties the first products with the label "CBR" were made commercially available, and bullish predictions have been made for the future of the technology. In this Report we present an overview of the state of the art of currently available commercial CBR tools.

Our objective in publishing the Report is to provide the reader with an informed insight into CBR technology so that he or she can better understand what are its capabilities and limitations. Of all the tools we evaluated, our view was that none was suited to being sold "shrink wrapped", except perhaps to an academic institution or an R&D laboratory. For this reason, we felt there was little value in a "PC-Expert"-like comparison of the tools and consequently have not produced such a rating. Instead, we have explained the main underlying principles of the tools and described their key features. Our aim has been to supply enough information to allow the reader to make a judgement as to which tool would be most suited to his or her CBR project, on the assumption that the work would be undertaken jointly with either the tool vendor or with a consultant.

This comparative study is original in that it is based on tests performed using the same data sets for each tool. We defined objective **evaluation criteria** and designed appropriate tests and test procedures in order to apply them. We chose **case bases** with properties that allowed us to test a wide range of features of the tools. Each tool was extensively tested on behalf of the authors of the Report by one member of a team of four Masters students at the University of Kaiserslautern (Germany) over a period of several weeks. We refer to this group of students (Harald Holz, Alexandre Meissonnier, Carsten Priebisch and Wolfgang Wilke) as the "Evaluation Team" elsewhere in the Report. Each student was solely responsible for one system (except for Wolfgang Wilke who evaluated two). The members of the Team entered into the spirit of the exercise by strongly advocating their particular tool(s) during discussions and in writing up the evaluations. All tools were tested on the same PC 486 DX2-66 with 16 Mb of RAM.

We asked all CBR tool vendors known to us at the time to support our evaluation by providing us with the most recent copies of their systems. By conducting our tests, we have determined the current state of the art of five currently-available CBR tools, and accumulated valuable information that can be used to guide future developments in CBR. We asked AcknoSoft S.A., Cognitive Systems, Inc., Esteem Corp., Inductive Solutions, Inc., Inference Corp., ISoft S.A. and tecInno GmbH for copies of their respective tools. We obtained PC versions of CBR EXPRESS, ESTEEM, KATE 3.0, REMIND and S^3-CASE. We also asked Lockheed for a copy of RECON, but the company was not interested in having its product evaluated since it is only sold in conjunction with a service contract and not as a "shrink wrapped" tool. We learned of the existence of "*The Easy Reasoner*" by The Haley Enterprise and of a few other CBR products only after it was too late for them to be included in the present edition of our Report. We hope that a future edition will enable us to include more tools and more recent versions of those we evaluated.

We used two main criteria to help in selecting test domains. The first was that we should all be able to understand them to facilitate the interpretation of results. The second criteria was to have at least one test domain that would be easily understood by everyone. For the first requirement, we chose *Fault Diagnosis of CNC Machining Centres*, a domain where we all had experience. We also chose *Identification of Marine Sponges* where we had the opportunity to involve an expert at the Museum of Natural History in Paris, and the TRAVEL AGENCY domain that met the second requirement of being widely comprehensible. In addition, we chose the CAR domain from the University of California at Irvine (see Appendix I) Repository of Machine Learning Databases because it is publicly available.

A major requirement of an evaluation is to produce results that are as objective as possible. Unfortunately, in some cases we had no objective experimental evidence that could bring out some of the interesting features of each tool. We therefore used a mixture of formally-defined tests that are reproducible, together with more qualitative evaluations that resulted from extensive discussions within the Evaluation Team.

The suppliers of all the tools evaluated had the opportunity to read a draft of this report and comment on it. Appropriate feedback has been included as footnotes. Furthermore, all known vendors of CBR tools were offered the opportunity of including summary details of their product in the Report. These details were obtained early in 1995 and the text has been supplied by the vendors. They appear in Appendix 2.

This work was carried out as a result of the INRECA European project in which both AcknoSoft and the University of Kaiserslautern are involved. We asked Ralph Barletta to join us, not just because of his experience in the field, but also because he is now an independent consultant.

Disclaimer

We wish to make it clear that, although we (the authors) defined the criteria for the evaluation tests, we were not involved in the actual testing procedure. The conclusions of Chapters 9, 10 and 11 reflect the opinions of the members of the Evaluation Team. In particular, Ralph Barletta (who led the development of ***REMIND*** *whilst he was employed at Cognitive Systems) and Michel Manago (who led the development of* ***KATE****) would like to point out that they were not always in agreement with the comments made about the respective tools.*

Executive Summary

Case-Based Reasoning (CBR) is a technology that solves problems by storing, retrieving and adapting past cases. CBR appeals to those professionals who solve problems by recalling what they did in similar situations. CBR first appeared in commercial tools in the early 1990s. Since then, it has been used to create numerous applications in a wide range of domains. These include: financial analysis, risk assessment, technical maintenance, process control, quality control, medical diagnosis, software support systems, forecasting, planning, design, classification of objects, photo-interpretation and real estate appraisal. Although the technology of CBR originated in AI laboratories where scientists studied Cognitive Psychology, it has now also become a technology for industrial and business applications.

Some of the characteristics of a domain that indicate whether a CBR approach would be suitable are:

- there exist records of previously solved problems;
- historical cases are viewed as an asset that ought to be preserved;
- if there is no case history, it is intuitively clear that remembering previous experiences would be useful;
- specialists talk about their domain by giving examples;
- experience is at least as valuable as textbook knowledge: CBR makes direct use of past experience.

The potential benefits of using CBR technology are:

- discovering knowledge in data;
- delivering consistent decisions throughout an organisation;
- preserving the know-how of the most talented specialists by capturing their experience;
- transferring experience from the skilled specialist to the novice;
- building a corporate memory by sharing individual experience.

Key issues when building a CBR system are:

- representing a "case" so as to capture its true meaning;
- indexing cases to retrieve them quickly;
- assessing the similarity between a current case and retrieved ones;
- adapting a solution that worked in the past to our new problem;
- integrating CBR into an organisation.

These issues imply that significant knowledge about the technology and skills in applying it are needed in order to build real-world applications.

CBR technology is not an alternative only to rule-based expert systems, but also to statistical data analysis, information retrieval, neural networks and even database languages. We summarise below how CBR can be compared to these techniques:

Rule-based expert systems: By not requiring specialists to describe their know-how as logical rules, the CBR approach overcomes what has historically been one of the main stumbling blocks in building

expert systems. CBR can be used to build, validate and maintain decision support systems. A CBR-based system can keep up with the knowledge that workers learn through their daily experience and can handle domains where there are many exceptions to rules and where problems are not fully understood.

Statistical data analysis: classical statistical methods and CBR are complementary. CBR does not rely heavily on assumptions about the data, such as its statistical significance, and the independence of the variables (attributes) that describe the data. Statistical methods are intended to infer characteristics of populations from those of individuals. CBR works by making decisions based on individual cases regardless of their statistical significance and can be used for exploratory analysis and data mining.

Neural networks: the main difference between CBR and neural networks is that a CBR system can justify its recommendations. A neural network is a black box and, as such, is appropriate to signal recognition and similar tasks. It is less appropriate to applications such as equipment fault diagnosis where the user needs to understand why the system has assigned a particular cause to a fault.

Relational databases: a common response to a CBR system is "I can do that with my relational database". However, CBR systems support fuzzy queries and retrieval and provide rich indexing support. Unlike a database query that retrieves exactly what has been requested, CBR retrieves cases that are similar in some sense. It uses background knowledge to identify hidden similarities (*e.g.* yellow and white are both light colours and are more similar than white and black).

Induction and machine learning: there is much confusion about the relationship between induction and CBR. Several shells offer facilities that are derived from work on induction. The difference between a pure inductive approach and a CBR one that uses induction to build indexing structures is not "what technology is used?" but rather "is the technology used in a way that supports CBR?". The answer to a CBR query is a set of cases that are similar.

Instead of a "PC-Expert" like comparative evaluation of the tools, we decided to present the underlying technology and focus on a higher level comparison that brings out the unique characteristics of each tool. Our purpose was not to rate the different tools but to give enough information for the reader to be able to judge for himself which tool is most adequate for his needs. We evaluated CBR EXPRESS (Inference), ESTEEM (Esteem Corp.), KATE (AcknoSoft), REMIND (Cognitive Systems) and S³-CASE (tecInno) using the same set of data on the same PC according to a carefully selected set of criteria.

The evaluation was structured around twenty technical and eight ergonomic criteria. These included:

- Assessing similarity.
- Representation and use of knowledge.
- Effects of noise and incompleteness of data.
- Performance and speed.
- Correctness, completeness, consistency and effectiveness.
- Degree of control for the developer.
- The extent to which the user interface can be customised.
- Explainability and modelling support.
- *etc.*

To help assess the tools against these criteria, we used eleven tests based on case data from four different domains: MARINE SPONGES, CAR, TRAVEL AGENCY and CNC MACHINE TOOLS. For criteria that could not be measured using objective tests, subjective tests were used. The comparative evaluation was undertaken by four students at the University of Kaiserslautern who worked independently of the authors of this Report.

In summary, the findings were:

- CBR EXPRESS appeared to be a tool that provides a number of standard CBR features combined with a comfortable user interface and surprisingly fast retrieval speed. It is very much focused on help desk applications. After performing the tests, we suspect that there may be an indexing mechanism that is hidden at the heart of the system but we were unable to reach a definite conclusion about this. If CBR EXPRESS does not have such an indexing mechanism, this is likely to prevent it from handling larger volumes of case data. If it does have one, the absence of information in the documentation and the lack of control by the user on when to reindex the case base makes it difficult to understand and test its advantages and limitations.

- ESTEEM's main advantages are: the possibility of using nested case structures and a rule mechanism that allows it to automatically adapt cases and compute similarity measures. Both of these features are inherited from its underlying Kappa-PC system. A weak point was a lack of an explicit mechanism for handling missing values. In general, the tool requires further improvements which, according to the supplier, are already being addressed.

- KATE offers a combination of induction and CBR. The CBR module allows dynamic indexing of cases and can handle problems such as unknown values during consultation. The induction module generates nodes with multiple branches (non-binary trees) and is resilient with respect to noise (*i.e.* errors and incomplete data) in the cases. We did not evaluate the nearest neighbour algorithm that is included in a later release of KATE. This was a weakness of the system we tested.

- REMIND offers a combination of different techniques (nearest neighbour and induction). The induction system generates binary trees using the CART algorithm. The key feature of REMIND is its ability to use background knowledge from experts to improve indexing, retrieval and similarity assessment. One weak point of REMIND is that it is a closed environment and that the case data cannot be exported.

- S^3-CASE is a young product with many features. It is very much tied to the SMALLTALK-80 programming environment which provides some benefits, but also adversely affects memory requirements, speed and the cost of runtime systems. S^3-CASE offers the power of a real programming language to customise features such as the similarity measure, the architecture of the system and the user interface, and it is fully portable across a wide variety of platforms.

A major point to come out of the evaluation process was that, although the basic principles of CBR may seem simple to understand, the underlying techniques are complex and it is the subtle details - never documented in the literature - that make all the difference. The description of the technology in the first part of this Report and the detailed discussions on the evaluation, strongly support this view.

One thing that became clear to us after performing this evaluation was that tools alone are not enough to field a CBR application. To make an analogy, buying a saw and a hammer will not turn you into a carpenter but if you do not have such tools, you will never become one. For all the applications that were reported as being in real use, a key factor of success was availability of "first class consultancy" around the tool.

Concerning the CBR market, the reality is that no vendor has yet sold a large enough volume to talk about the "CBR market" as one would talk about the "database market" or the "spreadsheet market". However, some forecasters predict an exponential growth of the market for CBR technology and the present situation may evolve in the near future.

We believe that a general trend for the future is to have systems that are more and more open in order to add CBR capabilities to existing software instead of having standalone tools.

To give a feel for the variety of problems to which CBR has been applied (pure CBR retrieval, induction-based retrieval or a combination of both), here is a sample of applications developed using the technology:

• American Airlines: technical support of the SABRE airline reservation system	• Sainsbury's: help desk for troubleshooting breakdown of sales terminals
• American Express: credit card risk assessment	• Lockheed: layout of composite materials in an autoclave (see Chapter 6)
• Andersen Consulting (Midwest insurance client): property and casualty underwriting	• Matra Space Corporation: satellite fault diagnosis
• ATT Bell: help desk	• Mitre: air traffic control
• Black & Decker: customer service hotline	• Mistubishi Electric Corp.: plant information management
• Blue Cross: medical diagnosis	• Microsoft: Intelligent user assistance bundled in *Windows*
• British Airways: maintenance of aircraft	• Naheola Mill: process control
• British Petroleum: gas-oil separation for an oil drilling platform	• NASA: process planning support, Space Shuttle landing decision support system
• Caledonian Paper: repair faults of electrical drives	• Nestlé: process control
• Cfm International: maintenance of aircraft engines	• Nippon Steel: process specification
• Compagnie Bancaire: credit assessment	• Philips: configuration of X-ray control systems

• Compaq: diagnosis of printers	• Prudential: life and motor insurance
• Daimler-Benz: off-line quality control of Mercedes gear boxes	• Roussel Uclaf: data analysis
• DEC: recovering from hard disk failures in the VMS operating system	• Shai: architectural/engineering
• Dun and Bradstreet: technical support and investment management	• Sepro Robotic: help desk for plastic injection press robots
• Elf Aquitaine: classification tasks	• Siemens: selection of synthetic materials
• French Ministry of Defence: command and control systems	• SINTEF: mud drilling for the oil industry
• General Dynamics: diagnosis of submarines	• SITA: telecommunication network management
• General Electric: maintenance of mission critical equipment's	• The French Institute of research in agronomy (INRA): tomato plant diagnosis
• General Motors: maintenance of cars	• Touche Ross: risk assessment
• GTE: health care, network traffic control and monitoring	• UK Department of Social Security: help desk for hardware and software
• Honeywell: training US Air Force pilots	• UK Electric: maintenance of electrical turbine generators
• Paris' Hospitals: epidemiology	• University Hospital Munich: medical diagnosis, personnel scheduling
• IBM: marketing program to assist in writing and marketing OS/2 applications	• VINITI: chemical safety
• IMS (for Coïllte): forestry management	• Volkswagen: quality insurance
• Institut Français du Pétrole: selection of lubricants	• Westinghouse: nuclear fuel refinement
• ITT Europe: in-process quality control of electronic circuits	

To conclude, we would like to recall the following saying "Data is a burden, information is an asset". CBR can potentially turn data into an asset.

Chapter 1: A Short History of Industrial CBR Tools

1.1 History of CBR Tools in the USA

Case-Based Reasoning (CBR) research in the USA was boosted by a three year Defence Advanced Research Project Agency (DARPA - now renamed as ARPA) initiative in CBR in 1987. The project's aim was to combine the work of several American universities that were carrying out research in CBR with those of a commercial company in order to blend the universities' research results into a generic CBR tool. The company was Cognitive Systems, Inc., headed at the time by Dr. Roger Schank, a pioneer in developing and promoting CBR technology as an alternative to other approaches.

By 1990, Cognitive Systems had developed a research prototype tool called "*The CBR Shell*" written in Common Lisp for Macintosh computers. This tool had facilities for representing, storing, indexing and retrieving cases. Applications in several domains were demonstrated, including battle planning, natural language understanding and telex message classification. After demonstrating the tool at several conferences and workshops in 1990, and having received a large degree of interest in it from both government and industry, Cognitive Systems began developing a commercial-strength, multi-platform version of "*The CBR Shell*" that would eventually be released in the spring of 1992 as REMIND.

As momentum began to build for the commercial use of CBR technology in 1990, other US companies began to develop CBR tools of various kinds. In mid-1991, Inference Corporation released a CBR-based help desk building tool called CBR EXPRESS. CBR EXPRESS had the unique ability to integrate natural language text into the case indexing and retrieval process, making it easier for end users to interact with the case base in the course of problem solving. In addition, CBR EXPRESS had a custom user interface that was specifically geared towards building help desk applications. CBR EXPRESS rapidly became a front runner in knowledge-based help desk tools with, according to the supplier, over 13,000 copies sold world-wide. Also in late 1991, ESTEEM was released, integrated into the **expert system** shell Kappa PC. This provided a system able to integrate CBR with rule-based knowledge. ESTEEM provided CBR capabilities to end users at a significantly lower price than CBR EXPRESS or REMIND but with less functionality and limited capabilities for handling reasonably large **databases**.

After the initial three CBR tools (REMIND, CBR EXPRESS and ESTEEM) were released, other tools for performing various CBR tasks emerged in the US, including:

- *The Easy Reasoner* from The Haley Enterprise, a tool for software developers that combines CBR EXPRESS-like natural language handling, with REMIND-like **induction** and ESTEEM-like ability to integrate with an expert system shell (ECLIPSE).
- *Induce-It* (renamed *Case-Power*), which provides simple inductive indexing from *Excel* spreadsheets for case representation and storage.

A list of some of the US companies that have developed applications using induction or CBR is included in the Executive Summary.

1.2 History of CBR Tools in Europe

In Europe, up until early 1990s, there was little work performed under the heading of "Case-Based Reasoning", although several groups worked with CBR technology under different names. Donald Michie, at the Turing Institute in Glasgow, pioneered the closely-related field of induction in the early 'eighties and reported on many impressive applications. In 1988, KATE-INDUCTION, developed by Michel Manago, was made available and was recommended for use by Texas Instruments. This included some CBR features, although the term was not used at that time.

As regards CBR tools developed under that title, three have been developed by European companies.

- The first proper commercial European CBR shell appeared in mid-1991. Initially called *CaseWork*, it was later renamed KATE-CBR and is now sold by AcknoSoft.
- A second tool, RECALL, appeared in August 1993 from another French company ISoft, which was founded in the early 1990s.
- The third tool, S³-CASE, also appeared in 1993. Developed by the German company tecInno, it benefited significantly from the INRECA project (see below).

European CBR research was boosted significantly by the three and a half year INRECA project (ESPRIT 6322) that started in May 1992 (see also Chapter 12 and Appendix 4). INRECA aims at integrating induction and CBR techniques for building decision support applications. As mentioned above, INRECA led to the development of S³-CASE and to some of the most recent developments in KATE 4.0. Another important CBR research project, which also started in 1992, is FABEL, funded by the German Ministry of Research. FABEL aims at integrating CBR with model-based approaches for design tasks. The cases consist of graphically-represented layout fragments from an architectural design domain.

It appears that the European CBR community now has a distinctive character in that the emphasis is on applications rather than on tools, and on the integration of CBR with other technologies.

Work on induction and CBR undertaken by European companies has let to many applications. Some of these have been listed in the Executive Summary.

Chapter 2: Issues in CBR

CBR is influenced by several different domains, in particular Cognitive Psychology, Machine Learning, Knowledge Engineering, Information Retrieval and Databases. In case-based methods, a new problem is solved by recognising its similarities to a specific known problem and then adapting the solution of the known problem to the new one. By way of contrast, other methods of problem solving derive solutions either from a general characterisation of a group of problems or by searching through an even more general body of knowledge (Bareiss,1989).

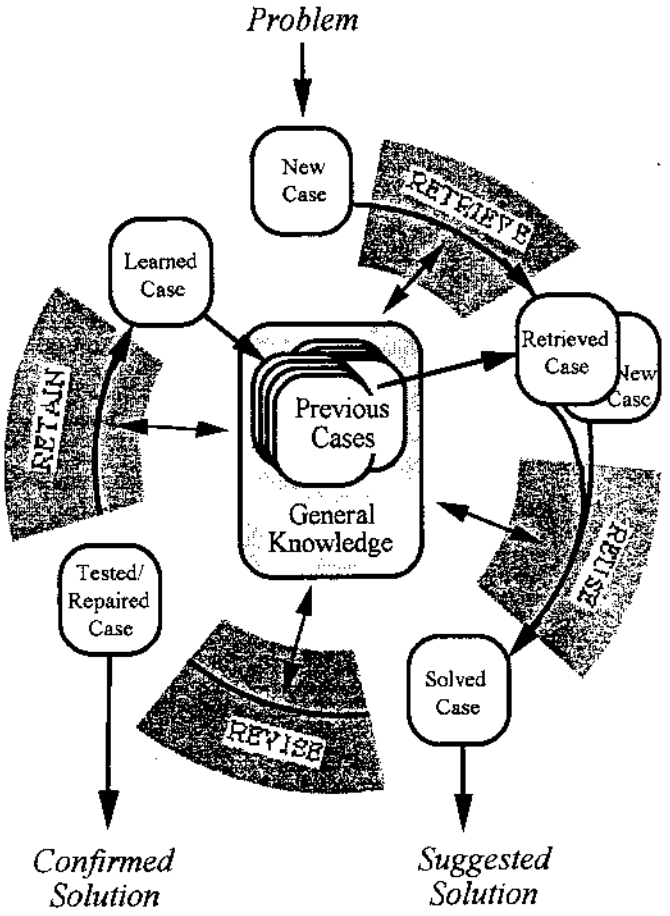

Figure 2.1. The Case-Based Reasoning Cycle (Aamodt & Plaza, 1994)

The Case-Based Reasoning cycle may be described using the following four steps:

- RETRIEVE the most similar case or cases;
- REUSE the knowledge about that case to solve the problem;
- REVISE and adapt the proposed solution to the new problem;
- RETAIN this experience for future problem solving.

In this chapter we discuss the important issues in CBR that are needed to achieve these four steps. These issues are: how to represent cases, how to define a **similarity measure** between cases, how to **index** them efficiently and how to retain, generalise and forget cases. Another important feature of the

above model is the idea of General Knowledge or **Background Knowledge**. This is knowledge about the domain that is not represented by individual cases. This idea will be further developed in a later section.

2.1 Case Representation

What exactly is meant by a "case" and *how* it is represented are two key issues in CBR. In some domains, a "case" is a specific problem that has previously been encountered and solved: the case base is the set of historical records of solved problems and the observed solution is empirically justified. In other domains, a "case" is an idealised, typical way of solving a problem or set of problems: the case base is a collection of hypothetical problems and corresponding solutions.

For both meanings of the term the way cases are represented or described raises important issues. In other words: What features characterise a case?; How are these features expressed (using numbers, symbols, objects etc.)?; What kind of case representation is needed (flat vectors, predicate calculus, frames or object-oriented languages)?

A case can be regarded as having three features:

- Its description;
- Its associated solution (called its diagnosis, its **class** or its target);
- The **justification** of its solution.

Justifications are an explicit representation of the problem solving process. They can vary in complexity and may be ignored or even skipped. For describing cases there is a wide variety of languages. We will not present each one in detail, but will instead provide an insight into their main characteristics.

2.1.1 Flat Representations

Let us suppose we want to describe an application such as the TRAVEL AGENCY domain (see section 8.2.4). For the sake of simplicity we restrict the description language to four **attributes**: *Hotel Name*, *Price*, *Region* and *Month*. Each attribute can a one of a list of possible values, depending on its type:

- *Hotel Name* is a string;
- *Price* is a number that varies between 50 and 300 ($);
- *Region* is a symbol with a finite set of values {Egypt, Tyrol, New York, India, ...};
- *Month* is an ordered attribute with values taken from {January, February, ..., December}.

A domain where each case is described by the same attributes is called a flat domain. It can easily be represented in a table where the columns contain values of the attributes and where each row corresponds to a case (for example, see Table 2.1).

However, this may lead to a lot of unnecessary information and to undesirable biases in the **similarity assessment** procedure.

Furthermore, suppose we wish to describe two cases: the first is a bungalow with one bedroom (*bedroom1*), the second is a bungalow with two bedrooms (*bedroom1* and *bedroom2*). In the first case, *bedroom1* contains a double bed, but in the second case, *bedroom1* contains a twin bed and *bedroom2* contains a double bed. If we were to use a non-structured model to handle the multiple-bedroom objects (see Table 2.2), it is possible that we would produce bad **indexing trees** that would fail to retrieve all the relevant cases, or wrong rules such as: "*If bedroom1 contains a double bed, then...*" whereas the right rule should be: "*If there exists a bedroom that contains a double bed, then...*".

	Bungalow		**Bedroom1**	**Bedroom2**	**Bedroom3**
	Kitchen	*Rooms*	*Type of bed*	*Type of bed*	*Type of bed*
Case 1	Equipped	1	double	Irrelevant	Irrelevant
Case 2	Equipped	2	twin	double	Irrelevant

Table 2.2. Flattening a structured representation

In fact, the first rule is consistent with the way we have represented the cases but fails to capture the internal meaning. It questions whether *bedroom1* contains a double bed, whereas what we want to know is whether there is a double bed in any bedroom.

This example demonstrates that the way cases are represented should be consistent with the way they are to be used internally.

Flat representations cannot adequately cope with information based on relations between various sub-parts: for this we need to go beyond representations based on propositional calculus and look to representations derived from first order logic. Some representations of this type are:

- Annotated predicate calculus;
- Conceptual graphs;
- Semantic nets;
- Frames and object-oriented languages.

The MARINE SPONGEs domain (see section 8.2.3) is best handled by a structured representation described by an object-oriented language. A flat language cannot capture the required complexity.

2.2 Assessing Similarity between Cases

An important step in the CBR process is computing the similarity between the new case those in the case base. A similarity measure should have the following properties.

- It should be reflexive: a case is always similar to itself.
- It should be symmetric: if case A is similar to case B, then case B is also similar to case A. In that sense, "similarity" is related to the notion of "distance".

The global distance between two cases is computed from the local distances in the attribute dimensions. Finally, similarity is not always transitive. That is, if A is similar to B and B is similar to C, we cannot always conclude that A is similar to C. For example, a white BMW is similar to a white Renault and a white Renault is similar to a red Renault, but the white BMW is not similar to red Renault.

2.2.1 Local Similarities

The overall evaluation of the similarity between two cases is based on the computation of local similarities between each attribute. The local similarity may vary, depending on the attributes' type or the size of the sets on which the similarity is computed. For instance, 10 is more similar to 20 if the size of the possible interval varies from 0 to 1000, than if it varies from 0 to 20. In Appendix 2, we present some frequently used local similarity measures.

Local similarities are generally defined on a restricted interval, for example, [0, 1]. This normalisation enables the user to combine them to evaluate global similarities (see section 2.2.2). Different local similarity measures can be used to cope with various data types. These measures are pre-defined in most of the CBR tools. However, it may be useful to define new local similarity measures, better suited to a specific domain. Some tools allow the user to define them through a programming language, or even to set up the similarity matrix directly between attribute values.

2.2.2 Global Similarities

Once a set of local similarities has been defined for each attribute, it is necessary to combine them in a global similarity measure. Hence, a global similarity SIM between two cases *A* and *B* described by *p* attributes, can be expressed by:

$$SIM(A, B) = F(Sim_1(a_1, b_1), Sim_2(a_2, b_2), ..., Sim_p(a_p, b_p)) \quad (1)$$

Where $F: [0, 1]^p \rightarrow [0, 1]$

Below are some global similarity measures; some of these are also used to describe distance metrics.

Global Similarities	Similarity Name
i. $SIM(A, B) = \frac{1}{p}\sum_{i=1}^{p} Sim_i(a_i, b_i)$	Block-City
ii. $SIM(A, B) = \sum_{i=1}^{p} \omega_i Sim_i(a_i, b_i)$	Weighted Block-City
iii. $SIM(A, B) = \frac{1}{p}\sum_{i=1}^{p} [Sim_i(a_i, b_i)]^2]^{1/2}$	Euclidean
iv. $SIM(A, B) = [\frac{1}{p}[\sum_{i=1}^{p} [Sim_i(a_i, b_i)]^r]^{1/r}$	Minkowski
v. $SIM(A, B) = [\sum_{i=1}^{p} \omega_i [Sim_i(a_i, b_i)]^r]^{1/r}$	Weighted Minkowski
vi. $SIM(A, B) = \max_i \omega_i Sim_i(a_i, b_i)$	Maximum

Table 2.3. Global Similarity Measures

where

- $p > 0, \omega_i > 0$ — p is the number of attributes, ω_i is a relevance weight ($\sum_{i=1}^{p} \omega_i = 1$);
- a_i (resp. b_i) — set of possible values of *A* (respectively *B*) for attribute i;
- Sim_i — local similarity measure as presented in previous section.

Global similarity measures are usually pre-defined in CBR tools: for instance, the default global similarity in REMIND is the Block-City. It is important to have the possibility of providing a weight matrix in which the values can be freely defined by the user, or automatically computed by the tool. In S³-CASE, the weight matrix computation can be achieved via a **learning** process.

Similarity assessment is a key part of reasoning with cases. However, no similarity measure is ever perfectly appropriate for all application domains. The best procedure is first to try the known similarity measures and then, if the results are not convincing, switch to more complex functions that are domain dependent. The determination of weights is also an important means of tuning a system where statistics, **knowledge acquisition** and tests can help achieve better results. It is therefore important to be able to define one's own similarity functions within the tools. Some tools offer the possibility of programming custom similarity measures.

2.3 Case Indexing

Case indexing is another important aspect of real-world CBR applications. The goal of case indexing is to select a subset of the attributes to be used in order to speed up retrieval. These attributes are generally organised in an index tree. A retrieval task starts with a new case and ends when the "best matching" case has been found. In theory, the better the case indexing mechanism, the faster the retrieval task ought to be.

There are two approaches to indexing: the computational approach and the representational approach.

In the computational approach the index is used to select a subset of cases that are then linearly scanned. There is no way of incorporating a method of using the index into the indexing mechanism. For example, in an index tree it is not possible to re-visit a node. The advantages of this approach are its simplicity and its speed of retrieval. Examples are the ID3 or CART algorithms described in sections 3.1.2 and 3.1.3.

In the representational approach, it is possible to include the method for its use in the index. For example, it is possible to back-track over previous nodes in an index tree. This should produce better quality results but can be slower when looking for a large number of similar cases. Two examples of this approach are presented below: a dynamic induction technique in section 3.1.5 and *k*-d trees in section 3.2.2 (section 2.3.2 describes how to build such a tree, whilst section 3.2.2 describes its use).

2.3.1 Indexing Options

Indexing options directly influence the way the cases are retrieved. An obvious first option is to have no index (linear retrieval). This option is available in most CBR tools and has a number of advantages:

- It works well on a small numbers of cases (the efficiency of the retrieval process depends on how complex the similarity function is and on how well it is programmed) and when most of the attributes are relevant;
- It guarantees to produce a list of cases ranked by their similarity to the new case;
- Attributes can directly be weighted to alter their relative importance.

The main drawback of this approach is that the retrieval time is linear: if it takes one second to scan a database of 1,000 cases, it will take two seconds for 2,000 cases, etc. If access to cases is slow (for example, cases are not stored in main memory and they have to be accessed over a local network), use of an index becomes necessary.

The second option is to build an index scheme where the most important attributes are organised in a tree structure (KATE, REMIND and S³-CASE all provide such a mechanism[1]). The most common index trees are **decision trees** (see section 3.1) and *k*-d trees (see next section and section 3.2.2). Use of these trees considerably speeds up the retrieval procedure by using only a small subset of the attributes: this leads to the selection of only a subset of cases on which similarity assessment is computed. However, it is important to allow back-tracking in the index scheme because of the danger of missing some similar cases that are not well indexed (see section 3.2.2).

The third option is to use background knowledge in the form of a set of "prototypes" or rules that guide the search towards specific cases (as in REMIND). Using this option, the retrieval process is similar to that of using an index tree. However, prototypes are directly defined by the user whereas index trees are created by the system.

Finally, some hybrid approaches combine different indexing schemes. For instance, it is possible to chain an index tree or a prototype with a linear retrieval approach. More complex systems in academic environments use a hierarchical discrimination network, based on cases and attributes, that evolves over time. For instance, the dynamic memory model, developed in Kolodner's CYRUS system (Kolodner, 1983), is based on a hierarchical structure of "episodic memory packets" or "generalised episodes". The case memory built in the PROTOS system (Bareiss, 1989) is embedded in a network structure of exemplar cases, indexes and categories.

The current CBR tools use only the three first options.

2.3.2 The *k*-d Trees Indexing Mechanism

In this section we present an indexing scheme, originally developed for databases, that speeds up case retrieval. It is used for example in S³-CASE. An index tree is used to pre-process the attribute values in such a way that the number of cases classified as "interesting" can be reduced. For case retrieval it operates like a fixed indexing structure. In this section, we show how a *k*-d tree is built. In section 3.2.2 we discuss how a *k*-d tree is used for case retrieval.

A *k*-d tree is a binary search tree that uses multiple attributes. The aim in building a *k*-d tree is to create a well-balanced binary structure that makes the search easier and faster.

---

1 Supplier's note: the current versions of CBR EXPRESS and ESTEEM also offer tree-like indexing.

Every node within the *k*-d tree represents a subset of the cases of the case base. The root node represents the whole case base. Every inner node partitions the set of cases represented by that node into two disjoint subsets, using a discriminating attribute (for details about how attributes are selected and cases are partitioned, see Auriol *et al.*, 1994).

Within the *k*-d tree an incremental best-match search is used to find a set of n most similar cases (nearest neighbours) within a set of N cases with k specified indexing attributes (see section 3.2.2). The search is guided by application-dependent similarity measures based on user-defined value ranges.

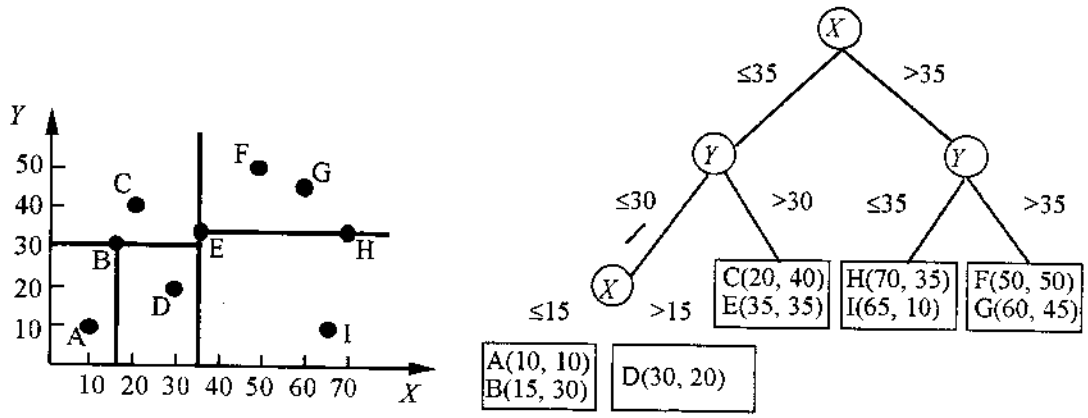

Figure 2.3. Example of a k-d tree

Figure 2.3 shows a set of nine cases (A to I) in a 2-dimensional attribute space (X and Y). The bold lines show how the set of cases is partitioned during the creation of the *k*-d tree and the resulting tree is shown to the right. The partitioning stops when the number of cases in a partition is below a user-specified threshold.

2.4 Retaining, Generalising, Adapting and Forgetting Cases

Retaining, generalising, adapting and forgetting facts are major functions of the human brain. Current AI techniques can seldom perform all of these at the same time. CBR tools can merely record new cases without generalising or forgetting them. Nevertheless, these topics are important for real-world applications and some academic prototype tools (*e.g.* PROTOS or CYRUS, see section 2.3.1) have been built to investigate how such features might be implemented. Note that in **CBR EXPRESS** it is possible to archive a case so that it is no longer taken into account during retrieval[1].

Depending on the application domain, adaptation may be necessary but none of today's CBR tools offer support for adaptation other than by explicitly programming the adaptation function.

---

[1] Supplier's note: KATE 4.0 also offers these features.

Chapter 3: Sample Algorithms

To implement a CBR system, a number of algorithms need to be used. As we have seen, these are required to enable quick retrieval, to build indexes and to assess case similarity. In this chapter we describe algorithms in all these categories: ID3 and CART for retrieval, dynamic induction and *k*-d trees for building indexes and IBL for assessing case similarity. We concentrate on the algorithms' main ideas and assumptions and show the risks involved in using an inappropriate algorithm.

A system is not necessarily classified as a CBR system on the basis of its underlying technology. Many algorithms at the heart of CBR systems have been inspired by work in statistics, information theory, databases and information retrieval. The key question is not so much "What is the underlying technology used by the system?", but rather "Is the technology used in a way that supports CBR activity?". Therefore, the choice of a CBR algorithm has to be guided by: an understanding of its underlying assumptions; how the CBR approach can be applied to the user's problem; what its strengths and limitations are; and by what occurs if the underlying assumptions do not hold in a specific context.

3.1 CBR and Induction

There is a lot of confusion about the relationship between induction and CBR. This stems from the fact that induction can be used in two ways:

- As a problem-solving technique, which assigns a new case to a class;
- As a means of building a tree which is then used as an indexing mechanism to a case base for case retrieval.

3.1.1 What is Induction?

Induction is a technology that generalises training cases (examples). It automatically extracts knowledge from cases in the form of a decision tree or a set of rules. This general knowledge (see Chapter 2) is then used to solve new problems. We distinguish between a pure inductive approach and a case-based one because induction first computes an abstraction of the case database (a decision tree or a set of rules) and then uses only this general knowledge to solve new problems. During the problem solving stage (for example, the **consultation** of a decision tree), the system behaves as if the case base no longer exists. On the other hand, CBR makes direct use of past experience (cases) during problem solving. Induction and CBR can both be viewed as an approach to developing experience-based expert systems. Induction *compiles* past experiences into knowledge that is then used to solve new problems. CBR directly *interprets* past cases in order to retrieve similar problems whose solutions are adapted to solve the current problem. We briefly introduce ID3 as an exemplary inductive algorithm (used at the heart of KATE-INDUCTION) and CART, the inductive clustering algorithm used in REMIND.

3.1.2 Inductive Approach to Learning from Cases: ID3

ID3 automatically builds a decision tree from a database of training cases. It uses a "greedy" search strategy (hill-climbing) and a heuristic to choose the most promising attribute. This heuristic is called "information gain" and is based on Shannon's entropy function (Quinlan, 1983). At each node in the decision tree ID3 evaluates information gain for all the attributes that are relevant and picks the one that is the most discriminating according to this heuristic.

For instance, let us consider the following subset of cases from the TRAVEL AGENCY case base, where the target attribute is the *Hotel name*.

	Hotel Name	Price	Region	Month
Case 1	Maharaja	125	India	April
Case 2	Cairo	200	Egypt	June
Case 3	Splendid	175	India	April
Case 4	Cairo	200	Egypt	May

Table 3.1. A Database of Hotels

An inductive decision tree built with ID3 could have the following form (Figure 3.1):

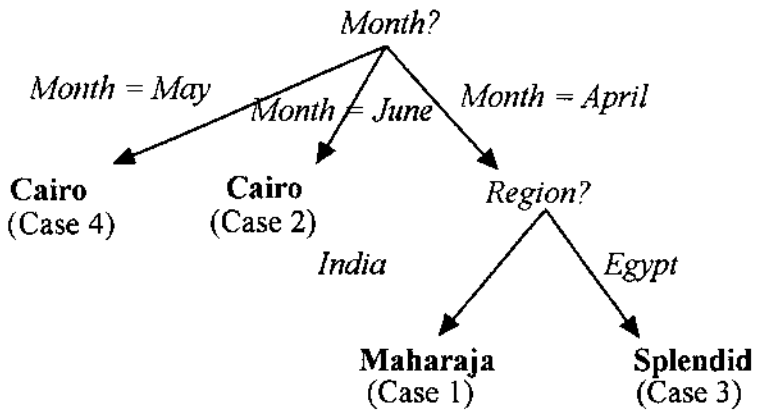

Figure 3.1. A Sample Decision Tree

The information gain measure estimates how well a specific attribute (Price, Region, Month) discriminates between the different classes (Cairo, Maharaja, Splendid). At each node, ID3 selects the attribute that yields the highest increase in the information gain measure. ID3 generates non-binary trees (for example, the *Month?* node has three branches in Figure 3.1). It is very efficient on large data sets (using its greedy search approach) and it generates decision trees that are well-balanced. This has advantages however the tree is used.

If we use it in a pure inductive way, the fact that it is well-balanced means few questions will be asked on average before reaching a conclusion. (A tree is used in a pure inductive way if we consider that internal nodes of the tree correspond to questions asked during a consultation and that leaf nodes

correspond to answers). If we use the tree as an indexing mechanism for CBR, the fact that it is well-balanced means that few tests will be made on average to identify relevant cases.

Some tools, like KATE-INDUCTION, use the basic ID3 framework but can handle complex data represented by structured objects with relations and can use background knowledge.

3.1.3 An Inductive Algorithm based on Statistics: CART

CART is an example of a CBR method that relies on a statistical approach, called the Bayesian approach, in order to build up indexing trees. The **induction tree** generator of REMIND uses such an approach when creating its indexing mechanism. This method is based on computing the "impurity" of a set of cases, according to a specific target attribute that partitions the set of cases. The impurity may be viewed as a measure of how heterogeneous the set of cases is in terms of balancing different subsets. When building the tree we choose, for each node, the attribute that maximises the "reduction of impurity" (*i.e.* the difference between the impurity of a "father" node and that of its "children"). Unlike the information gain measure used in ID3, this approach relies heavily on statistical data analysis: the reduction of impurity is equivalent to the distance of Kolmogorov-Smirnov used in statistics for creating segmentation trees. The usual restrictions when using a statistical method have to be taken into account.

The independence assumption for the attributes is particularly important. Bayes' theorem states that the probability Prob(H / $A_1 \wedge ... \wedge A_n$) of having a hypothesis H verified when observing the values of a set of attributes ($A_1 \wedge ... \wedge A_n$), is equal to the ratio of the probability of observing H and ($A_1 \wedge ... \wedge A_n$) simultaneously, with the probability of observing ($A_1 \wedge ... \wedge A_n$):

$$\text{Prob}(H \mid (A_1 \wedge ... \wedge A_n)) = \frac{\text{Prob}(H \& (A_1 \wedge ... \wedge A_n))}{\text{Prob}(A_1 \wedge ... \wedge A_n)}$$

As the information regarding Prob(H & ($A_1 \wedge ... \wedge A_n$)) may be very hard to come by, a more useful form of Bayes' theorem is:

$$\text{Prob}(H \mid (A_1 \wedge ... \wedge A_n)) = \text{Prob}(H) \times \frac{\text{Prob}((H \& (A_1 \wedge ... \wedge A_n)) \mid H)}{\text{Prob}(A_1 \wedge ... \wedge A_n)}$$

Unfortunately, the computation of Prob($A_1 \wedge ... \wedge A_n$) requires the estimation of the conditional probabilities of all subsets of attributes given a hypothesis - a number that grows exponentially. If all the attributes were independent - *i.e.* if for any attribute i, Prob(A_i) = Prob($A_i \mid A_j$) - we could make use of the fact that Prob($A_1 \wedge ... \wedge A_n$) = Prob($A_1$) × ... × Prob($A_n$).

Although the independence between attributes is a fundamental prerequisite for using the technique, it is often overlooked by those who provide and use AI tools. The statistical significance of the case base and the independence of attributes are also major restrictions of this approach. These assumptions seldom hold in real life domains.

Some implementations of CART, such as the one used in REMIND, generate binary trees where all tests are of the form "*Region* = India (yes/no)" "*Region* = Egypt (yes/no)", etc. However, the basic CART algorithm can be extended - with a corresponding high computational cost - to produce non-binary trees of the same form as ID3: "*Region* = ? (India/Egypt/...)".

3.1.4 Limitations of Induction Compared to CBR

Pure induction presents some limitations for building decision support systems that need to handle missing values during consultation: information about examples is generalised away during the induction phase and is no longer available during consultation.

For instance, let us consider the subset of cases described in Table 3.1 and consider the consultation of the tree in Figure 3.1. If the user answers "unknown" to the first question concerning the *month* and then "India" to the question about the *Region*, the result provided by the tree will be "Maharaja" with a probability of one third and "Cairo" with a probability of two thirds. However, if we consider the cases at the "Cairo" leaf nodes (Cases 2 and 4), we note that the attribute *Region* has the value "Egypt" unlike the current case where it is "India". Therefore, the current case is closer to the "Maharaja" hotel than to the "Cairo" hotel and the correct conclusion ought to have been "Maharaja" with a probability of one. Unfortunately, the information about the *Region* of Cases 2 and 4 has been generalised away during the induction phase and this information is no longer available during consultation.

This problem is not caused by the particular induction technique that we have used - we could have used another induction algorithm or even built the tree by hand. It is not caused by the decision tree formalism - we could have used rules (even chaining rules) instead of a tree. This particular problem is due to the fact that we are reasoning using *abstract knowledge* and have lost some useful information that was originally contained in the training cases. It is therefore a flaw of the knowledge-based approach to problem solving (reasoning about a problem using abstract knowledge). In order to provide a general solution to this problem, we must adopt a "pure" CBR approach that does not use a fixed indexing mechanism. Some CBR-like systems that use fixed decision trees to index the cases suffer from the same pitfall.

In section 3.1.5 we describe a technique for indexing cases dynamically. As with the inductive approach, case identification is based on a minimal number of attributes. Other CBR techniques, based for instance on nearest neighbours (see section 3.2), use the full case description instead of a minimal subset to assess the class of the case. The best approach depends on the application in question.

If there is a high cost associated with answering questions (for example, in order to perform some test on a piece of equipment, you have to take it apart), you are clearly interested in minimising the number of questions. If on the other hand you obtain all the information from sensors and the cost of answering questions is irrelevant, there is no overhead in answering all the questions and the only issue is the efficiency of the nearest neighbour matching.

3.1.5 A CBR Approach based on Dynamic Inductive Techniques

An interactive approach can be used for dynamically indexing cases incrementally. Instead of building the overall decision tree and then forgetting the cases (as with a pure inductive approach), we can use the same technique to build a single path, step by step, based on the answers supplied by the user during the consultation. For example, KATE-CBR uses the information gain criterion to determine the best attribute to be chosen at a particular level, but it develops only the subpart that corresponds to the answer supplied by the user. If the user answers "unknown", it selects the next best one and so on. The number of cases indexed in this path decreases quickly until a sufficiently similar case is found (or until the user decides to abort the consultation).

Let us use once again use the case base of Figure 3.1. According to the entropy criterion, the best attribute to be answered is *Month*. If the user answers "unknown", the next best attribute is chosen, for example *Price* (with a threshold of 150 that is automatically computed by the algorithm), until the user supplies an answer. Suppose the user answers 130, we can conclude that the hotel is "Maharaja". Only a single branch of the tree is developed and is used as an index for the next new case. This allows the system to dynamically index the cases based on the information available at run time.

3.2 CBR Methods based on Nearest Neighbours

Most practical CBR methods rely heavily on the standard nearest neighbours' algorithm. The K-nearest-neighbours algorithm is well-known in the field of statistics (classification and non-parametric discriminant analysis). Suppose we have a set of training cases and the value of the target attribute C_l (the class) is known ($l = 1, ..., L$). The statistical approach aims at evaluating the probability $Prob(C_l \mid X)$ that a new case X belongs to class C_l with respect to the classes of the cases in its neighbourhood. Bayes' theorem states:

$$Prob(C_l \mid X) = \frac{f_l(X)\,Prob(C_l)}{f(X)}$$

where $Prob(C_l)$ is the *a priori* probability of class C_l in the case base;
$f_l(X)$ is the density function of class C_l in the neighbourhood of X;
$f(X)$ is the density function of the case base in the neighbourhood of X.

Pragmatically, the size of the neighbourhood is given by the number of cases that belong to it (*i.e.* K, from where the algorithm derives it name) and the density functions are estimated by frequencies. The probability is then simplified to:

$$Prob(C_l \mid X) \approx \frac{n_l}{K}$$

where n_l is the number of cases among K that belong to class C_l.

The new case X is assigned to the class C_l that has the highest probability.

In practice, the computation of the K-nearest neighbours requires the evaluation of the similarity of X with each case if the case base. Section 3.2.1 describes the algorithm IBL that implements the basic ideas of the K-nearest neighbours algorithm.

If there are many cases, a linear scan of the case base is inefficient. Various case indexing mechanisms (see section 2.3) may be used for limiting the search space. In section 3.2.2 we explain how the indexing mechanism of *k*-d trees can be used to reduce the number of accesses to the case base.

3.2.1 A Sample Nearest Neighbour-Based Algorithm: IBL

IBL belongs to the class of clustering algorithms. It learns how to divide a set of examples into different categories. It can be characterised by the following three functions:

1. *Similarity function*. This function computes a numerical value for the similarity between a case of the **training set** and a new case;
2. *Classification function*. This function interprets the results of the similarity function and computes the class of the new case;
3. *Concept description updater*. This function decides which cases have to be inserted in the concept description and updates the classification records of cases in the concept description.

The basic similarity function used is:

$$Similarity(x, y) = -\sqrt{\sum_{i=1}^{p} f(x_i, y_i)} \tag{5}$$

where cases are described by p attributes and

$$f(x_i, y_i) = \begin{cases} (x_i - y_i)^2 & \text{if } x_i,\ y_i \text{ are numeric} \\ (x_i \neq y_i) & \text{if } x_i,\ y_i \text{ are symbolic} \end{cases}$$

IBL can be described in the following algorithmic notation:

Classified Data = Ø
for each Case x ∈ Case base do
1. for each y ∈ Classified Data do
   Sim[y] = Similarity(y, x)
2. $y_{max} = (y_1, ..., y_K)$ such that $Sim[y_k] = max$ (K-nearest neighbours)
3. if class(y_{max}) = class(x)
   then classification is correct
   Classified Data = Classified Data ∪ {x}
   else classification is incorrect

Other versions of the algorithm include more balanced similarity functions, attribute weights, filter on noisy data, etc. But the basic advantages of this algorithm remain the same:

- It allows incremental learning;
- It can learn incomplete concepts.

On the other hand, there are some drawbacks with the K-nearest neighbours method:

- It makes no generalisation of knowledge;
- It requires that the cases be represented as a flat table;
- Each new case needs to be compared with all cases of the database.

3.2.2 Retrieval with Indexes in *k*-d Trees

In practice, the last drawback is often the one that makes this approach inefficient. When cases are stored on a hard disk, or when they have to be accessed over a local network, the time needed for loading each case becomes significant. The number of accesses to the case base becomes a key factor affecting system performance. The use of indexes avoids the need to scan the whole case base during

retrieval and thereby improves the overall CBR system. As already stated in section 2.3, most CBR systems provide an indexing scheme. In this section we focus on the *k*-d tree retrieval mechanism.

A *k*-d tree of all the cases is traversed recursively from its root to look for the K nearest neighbours of a new case. The inputs of the search are:

- A query (new case) specifying the values of *q* attributes;
- A number K of cases to be retrieved, or a minimal similarity threshold s_{min};
- A *k*-d tree, represented by its root.

During the search, a priority list is maintained that, at any given time, contains an ordered list of the current K most similar cases and their similarity to the new case. The recursive search procedure (beginning with the root node) is bounded by two tests:

- If the current node is a final one, the priority list is modified according to how similar the new case is to the cases belonging to this node.
- If the current node is not a final one, the procedure is iterated on the child node specified by the value of the attribute of the new case for the current question.
- A Bounds-Overlap-Ball (BOB) Boolean test is then made to determine whether it is reasonable to inspect the other child nodes. If this test is false, the partition of the other child nodes cannot contain any K-nearest neighbours with respect to the new case and they are not examined further. If this test is true, the procedure is iterated on these nodes.
- At the end of the procedure, a Ball-Within-Bounds (BWB) Boolean test checks whether or not all the K nearest neighbours have been found. If it is false, the search is extended to previous nodes in the tree.

BOB and BWB are relatively simple geometrical procedures. They allow the system to test whether or not a given node may provide candidates for the K-nearest neighbours list, *without computing the similarities with all possible cases*. This relies on the following two main ideas.

- The query (new case) X_q is considered as the centre of a q-dimensional ball, whose radius is exactly given such that the K^{th} neighbour is located on the surface of the ball. All the cases of the priority list are located within this ball.
- A case belonging to the data space is a candidate only if it is located within the ball (that is, if its similarity with X_q is greater than the similarity between X_q and the K^{th} neighbour).

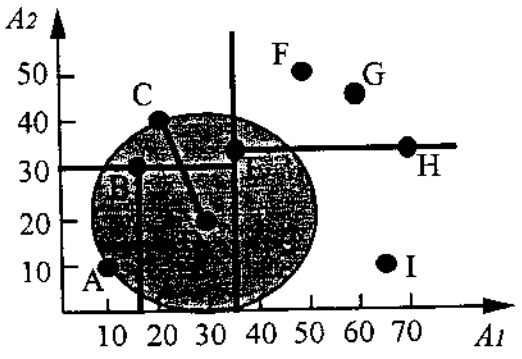

Figure 3.2. Ideas for the Bound-test with 5 cases

Figure 3.2 displays the bound test geometrical procedure for the new case D. The ball size can be defined in two different ways: *directly* through a pre-defined user threshold or *indirectly* by setting a minimal number of cases to be retrieved.

BOB test

In order to recognise whether a node of the tree is "of interest" (*i.e.* whether it may contain some candidates), the geometrical bounds of the node are used to define a *test point* in the dimension currently investigated. If this test point is in the ball, then the ball overlaps with the node and there may be a candidate for the priority list in this node. This search is performed by the BOB test.

The example below shows that X_{min1} belonging to the node K_1 is in the ball (and it may, therefore, be of interest to explore this node), but X_{min2} and X_{min3} do not belong to it.

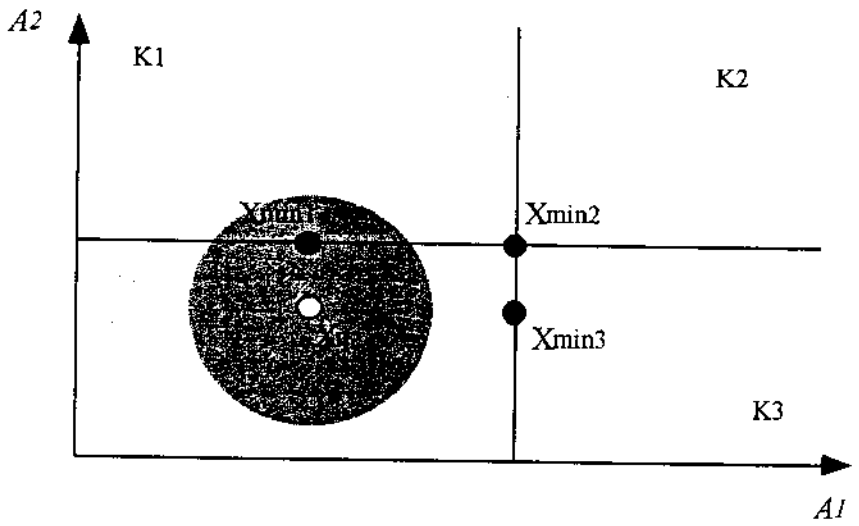

Figure 3.3. BOB Test

BWB test

The goal of the BWB procedure is to decide whether or not all K-nearest neighbours have been found. This test is made during the search procedure each time a node of the *k*-d tree has been investigated. The question that needs to be answered is: does the ball around X_q lie completely within the geometric bounds of already explored nodes (let us call this set the "bounding box")? If the answer is positive, it means that no case that lies outside the bounding box can have a better similarity with X_q than the cases already stored in the priority list. When the search is finished, its results are stored in the priority list.

To carry out this test, we have to verify whether or not the bounding box intersects the ball. This verification can easily be performed with simple geometrical tests.

Figure 3.4 shows a successful BWB-test in a two-dimensional space.

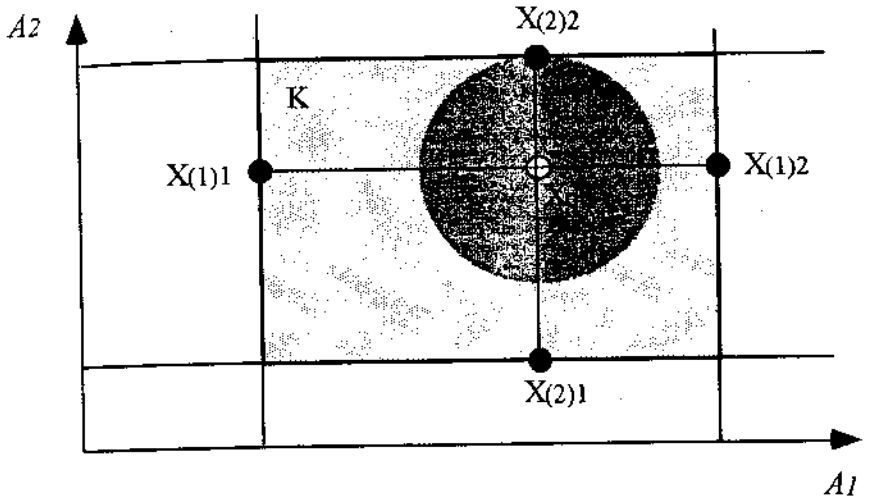

Figure 3.4. BWB Test

Final Remarks about the Indexing Mechanism Problem

Using a retrieval mechanism based on indexes can significantly decrease the number of cases that have to be scanned during the K-nearest neighbours process. However, we must keep in mind that a significant amount of time can be spent in building the indexing schemes and in implementing the retrieval mechanisms. Secondly, the efficiency of the retrieval mechanism depends heavily on the attribute types. For instance, the *k*-d tree mechanism cannot efficiently handle symbolic attributes. Finally, the time spent by the mechanism itself may be significant if many tests have to be performed (for instance, if the new case is poorly defined).

The best way to deal with the indexing problem is probably to start without any indexing mechanism and to test whether the system developed can scan the entire case base in a reasonable time. If cannot, study the types of the attributes and develop a custom indexing scheme. For instance, a causal model for symbolic attributes, or an indexing tree for numeric ones. If the database is only accessible through a network, it is usually necessary to develop a retrieval structure.

Chapter 4: Positioning CBR

In this Chapter we compare and contrast CBR with techniques derived from information retrieval, statistics, pattern recognition, knowledge-based systems, machine learning and neural networks.

4.1 Positioning CBR and Statistics

4.1.1 CBR and Linear Discriminant Analysis

A comparative study between linear discriminant analysis and CBR was conducted at Daimler-Benz on a gearbox quality control problem (Hübert & Nakhaeizadeih, 1993). There were 7,080 cases in the case base, described by 56 attributes. These cases were partitioned into 91 different classes (diagnoses). The results were evaluated through a cross-validation procedure where 80% of the case base was used for learning and the remaining fifth for testing. By selecting different test and learning sets randomly each time, Daimler-Benz calculated the average of the results for five different tests. The following table summarises the results in terms of the average percent of good results.

Tests	Case-Based Learning Algorithm	Linear Discriminant Analysis
1	93.4	61
2	93.4	60
3	92.6	62
4	93.2	61
5	93.5	62
Average	93.2	61

Table 4.1. Comparison between Case-Based Learning and Linear Discriminant Analysis

The statistical method performed poorly compared to the case-based learning method. Linear discriminant analysis requires a sizeable amount of data for correct operation (for example, to estimate the median of each class and the global covariance matrix). Theoretically, for 56 attributes and 91 classes, 1,603 parameters have to be computed. Even with a large number of cases, all these parameters cannot independently be estimated. Hence, the linear discriminant function is biased towards the first-estimated parameters.

4.1.2 Induction and Credit Scoring

A comparative study between an inductive approach (implemented using KATE-INDUCTION), a statistical data analysis method (credit scoring implemented using the SAS language) and a rule-based expert system (created by interviewing experts) was conducted for a credit assessment application (Perray, 1990). The case base contained 735 cases described by 40 numeric and symbolic attributes. The problem was complex for a statistical tool because, on average, 80% of the attributes had unknown

values. The final evaluation was conducted by the end-user with 300 new cases that were not presented during the induction phase.

	Experts	Induction	Rule-based expert system	Statistics
Perfect (%)	91	70.3	73.2	48.6
Near (%)	8.1	21.7	12.3	26.8
Total Good (%)	99.1	92	85.5	75.4
Delay (days)		14	60	25 [1]

Table 4.2. Comparative Study between Statistics and CBR

"Perfect" means that the answer was exactly the same as the one provided by four financial experts. "Near" means that the right answer was provided with a probability above 0.6. Compared to statistics, induction produced significantly better results. The knowledge extracted from the data was explicit and could be evaluated by field experts. The reasons why credit is granted - or denied - can be justified by showing the corresponding rules that have been extracted from the data or the cases that have been retrieved. Compared to the expert system, induction enabled delivery of the application in significantly less time.

4.1.3 Comparison between CBR and Statistics

In the previous two sub-sections we presented applications where CBR and induction-based techniques performed significantly better than statistical methods. However, we should not jump to the conclusion that CBR is always better than statistics. Statistics and CBR are complementary techniques in many problem-solving processes. Statistics works well on large amounts of standardised data to test known hypotheses. However, most statistical methods are not suited for exploratory analysis (*i.e.* when all hypotheses are not yet known) because they require strong underlying critical assumptions that are often overlooked by the end-user (for example, the independence of attributes).

In addition, when using statistical methods, it is hard to take into account common sense or background knowledge. CBR on the other hand can make use of background knowledge when available since it integrates numeric as well as symbolic techniques.

4.2 Comparison with Information Retrieval

Although CBR and Information Retrieval (IR) have a great deal in common and are often used for similar tasks, there are relatively few studies comparing the two techniques. This may be due to the fact that CBR and IR originated from - and were developed in - two different communities.

CBR, like IR, focuses on retrieving relevant information from a database (case base) of collected data. Both allow flexible database querying and result in a collection of relevant but inexact matches: this is

[1] This includes tuning the credit scoring functions using the SAS language.

the main difference between the two technologies and relational database systems). CBR and IR differ in the following ways:

- *Data type*. Whereas IR methods mainly operate on textual data, traditional CBR methods operate on vectors of several basic data types: real, integer, symbol, Boolean, string, etc.
- *Amount of data*. IR methods can handle huge amounts of data. IR can search thousands of documents, consuming gigabytes of memory. CBR systems are comparatively more limited.
- *Use of knowledge*. IR systems operate without knowledge of the user's problem-solving task. They provide a generic indexing and retrieval engine that can be used for a wide range of tasks. As a consequence of this they have limited accuracy for any given query. CBR systems, on the other hand, make use of knowledge about the problem-solving process in order to build effective indexes, such as decision trees or *k*-d trees and to improve retrieval accuracy.

These difference are true in a shallow comparison of the CBR and IR systems. However, in a deeper comparison, the differences become blurred. Current IR tools often operate on mixed data types and use a thesaurus or concept hierarchies during retrieval. On the other hand, some commercial CBR tools do not represent and use background knowledge. They often only use a similarity function on flat attribute-value vectors.

Therefore, it is better to say that CBR tools are primarily concerned with mixed representations whereas IR systems are primarily concerned with textual databases. Furthermore, CBR tools often explicitly represent the knowledge they use, whereas IR systems do not. Hence, it is possible to consider that, for complex-structured application tasks that require an integration of different, knowledge-intensive problem solving and learning methods (*e.g.* in synthetic applications domains), the difference between CBR and information retrieval becomes very apparent. However, for application tasks such as decision support, help-desk systems and diagnosis, where syntactic approaches to similarity assessment and simple reuse strategies are often sufficient, the differences between a knowledge-based IR system and a low-level CBR tool are minor, especially when compared to knowledge-based approaches to information retrieval.

4.3 Comparison with Rule-Based Expert Systems

Developing rule-based expert systems that can solve complex real world problems is a difficult task. One of the main difficulties is due to the fact that rules have to be provided by human experts. Human experts are very good at solving practical problems, but not so gifted at explaining *how* they have solved a particular problem. In addition, they can seldom articulate this knowledge using logical rules that can be expressed in a formal language. This problem is known as the "Knowledge Acquisition Bottleneck". Another problem with classical expert systems is maintaining this knowledge over time.

CBR and inductive reasoning provide methodologies for building, validating and maintaining applications. Instead of providing rules, specialists talk about their domain by giving examples; it is more intuitive to answer a question such as "*Have I ever seen this problem before?*" than to provide a general definition of a class of problems! Rules handle big chunks of the problem domain well but perform poorly on boundary regions where experience needs to be accumulated on a daily basis. CBR is valuable when problems are not fully understood (weak models with little background knowledge

available) and where there are many exceptions to the rules. In such situations the number of special or subtle contexts make a rule-based approach inadequate. A typical example of a purely case-based approach in a weak domain is the *Clavier* application described in Chapter 6.

Finally, methods based on cases are incremental. They can learn from experience and keep up with the knowledge that workers acquire in their daily experience. This maintenance task is much more difficult when using a **rule-based system**: as the system expands, smaller and smaller chunks of the domain have to be incrementally covered by rules. This results in declining productivity, increasing difficulty in maintaining the rule base and, ultimately, leads to an incomplete coverage of the problem.

The following table gives a rough classification of methods that use cases and/or rules.

	Exact Matching	**Partial matching**
RULES	Standard Rule-Based Approach	Analogy-Based Approach
CASES	Standard Database Approach	Case-Based Approach

Table 4.3. Matching of Cases versus Matching of Rules

According to the complexity and/or the goal of the task to be achieved, it is possible to evolve smoothly from a rule-based system where all the possibilities of decision making are sketched, to a purely incremental case-based system where cases model the entire problem domain. A helpful and promising compromise can be found in hybrid systems where the domain is modelled with rules as far as possible and boundary regions are handled by cases.

4.4 Comparison with Classical Machine Learning Approach

As regards machine learning, CBR is not as well understood as inductive learning. There is no general consensus about the overall learning task that is addressed by CBR. A major distinction is that machine learning systems make a strong separation between learning and problem solving. Learning involves analysing training examples to extract functions or rules; problem solving involves applying these functions to new incoming problems. In contrast, CBR does not separate the two. However, the transformation of a simple learning algorithm into an equivalent case-based variant underlines in principle the equivalence of symbolic and case-based methods. The equality of the learning power of symbolic and case-based classifiers is even proven for the area of inductive inference. The proof is based on the learner's ability to adjust the similarity measure to the given problem. There is no theoretical difference between case-based classification compared to the traditional symbolic learning approach in this simple framework. Both mechanisms can learn the same concepts. Nevertheless, work in CBR is concerned with complete systems, whereas work in symbolic machine learning is more concerned with algorithms. In addition, CBR explicitly includes the notion of memory which eases the mapping onto practical problems.

In practice, an important difference between case-based and symbolic classification algorithms concerns the representation of the learned concept. The symbolic approach corresponds to a kind of compilation process whereas the case-based approach may be viewed as a kind of interpretation during runtime.

Which approach should be used in any given situation depends mainly on the simplicity and adequacy of the representation of the given knowledge to the application to be delivered.

4.5 Comparison with Neural Networks

Neural nets perform better than CBR in a knowledge-poor environment when the data cannot be represented symbolically, as in radar signal recognition. Their field of application also extends to pattern recognition where there are many points of raw data, as in vision, speech and image processing. Neural nets are very resilient to noise during the consultation phase: for instance, even with only a fraction of the original attributes having values, retrieval performance can still be very high. However, neural nets have drawbacks that have yet to be removed. Firstly, they are not suitable when background domain knowledge has to be taken into account. Secondly, they cannot cope with complex structured data. Furthermore, in order to perform well - efficient training and good convergence - the coverage of the domain has to be exhaustive during the "learning" phase.

Ergonomically, neural networks suffer from a lack of transparency. Users cannot judge the validity of a network's decisions because of the nature of its inner workings: the output of the network is a function of weighted vectors that depends on the network's architecture and the learning mode used. No justification or explanation of its output can be easily provided. On the one hand, the classic bottleneck problem of questioning field specialists is avoided (as is the case with other machine learning paradigms); on the other hand, the experts can neither validate nor modify the resulting system.

Neural networks and CBR can be integrated within a larger a system. For instance, in the PATDEX system (Althoff & Wess, 1991), the relevance of the attributes with respect to a given diagnosis is updated through a competitive learning mechanism derived from unsupervised neural networks. The technologies also complement each other in that each is suited to different types of application. Depending on the amount of available knowledge and on the goal of the target application, the appropriate technology can range from a low-level knowledge approach where little explanation is required (neural nets), towards a rich and complex learning scheme that involves high-level learning techniques.

Chapter 5: Characteristics of Application Domains

5.1 Introduction

In this section we provide a classification of CBR applications. We first divide CBR applications into two basic task types: classification-oriented tasks and planning/synthesis-oriented tasks (Figure 5.1).

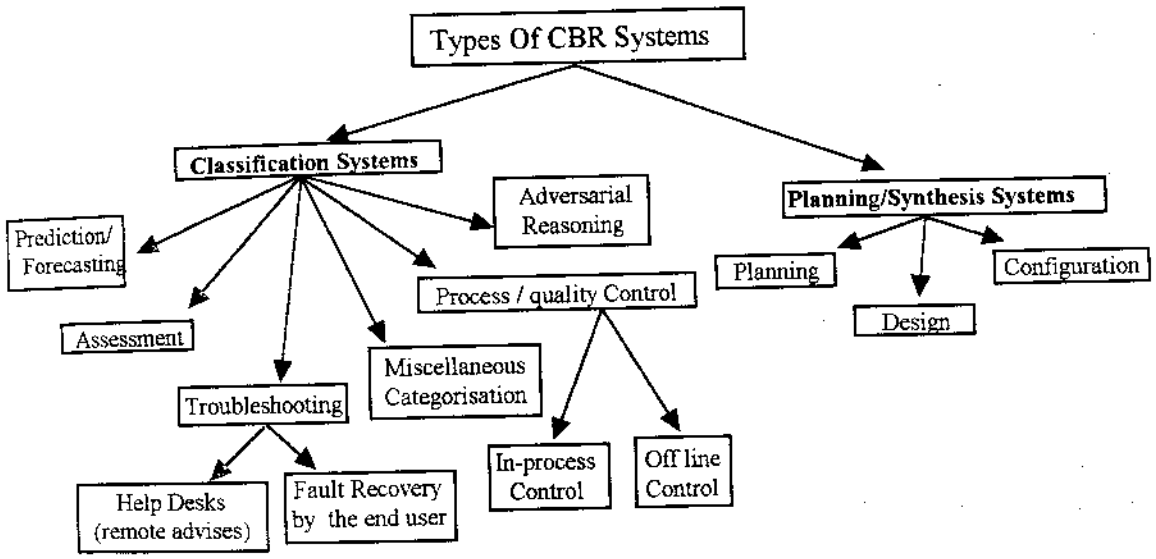

Figure 5.1. Types of CBR Systems

Classification-oriented CBR systems cover a wide range of applications. They operate as follows: cases that match the new case are retrieved from the historical case base and the new case is classified as being in the same 'class' as the best-matching case. Most commercial CBR tools are geared towards solving classification tasks because they focus on the case retrieval process.

The following are examples of classification systems that have been developed using CBR:

- Prediction/forecasting: predictive maintenance and stock market analysis.
- Assessment: risk analysis (banking, insurance), real estate estimation, appraisal, legal/insurance claim determination
- Help-desks: after-sale support of equipment (computers, jet engines, appliances) and software hotline support (*e.g.* the SABRE system)
- Fault recovery: diagnostic and decision support systems (*e.g.* on-line troubleshooting of a CNC machine

- Process control: on-line control (*e.g.* oven temperature control, optimisation of phosphate coating for steel wires), off-line quality control (*e.g.* identification of poorly manufactured transmission systems for cars)

Planning/synthesis-oriented CBR systems attempt to create an artefact (*e.g.* a design, plan, configuration, schedule, etc.) by using cases of previously-created artefacts as a guide or template for creating the new one. In planning/synthesis-oriented CBR systems, the adaptation process that follows the retrieved step requires much more effort than in classification-oriented systems. In fact, in most cases, the adaptation is highly-dependent on the application domain. To design a bridge, for example, we do not adapt a bridge that is 100 m long to one that is 80 m long in the same way as a worker in a factory changes the layout of composite parts that go into an autoclave.

5.1 Classification Tasks

Classification tasks are ubiquitous in business and industry. They are characterised by a need to place an object into a predetermined category. The following common business questions characterise examples of classification tasks:

- Is this person a good loan/insurance risk?
- How much is this house worth?
- What is causing this problem?
- Who is going to win this case?
- What is the diagnosis for this person's symptoms?
- Is this claim legitimate?
- Is this person a potential buyer of my product/service?
- Is/Are there oil/gas/minerals on this tract of land?
- Should the temperature be increased or decreased to keep the process within tolerance?

Despite being very different, all these questions can be answered through a process of classification. Using a case-based approach to problem solving, classification becomes a process of retrieving the best matching case and using the category of that best case (or best cases) as the category for the new situation. The overwhelming majority of operational CBR applications to date are classification systems. This is true for several reasons:

- *Cases tend to be simpler to represent and easier to collect.* Typically, one of the case's attributes is the "outcome" or "class". The "class" attribute is the label of the stored case and is used to determine the classification of a new incoming case. Because the representation is simpler, more cases can usually be collected, which greatly improves the coverage and classifying accuracy of the system.
- *A wide range of robust indexing/retrieval schemes applies.* Since cases in a classification-oriented system are often represented as a flat attribute vector with a single class attribute, a wide range of time-tested, mathematically sound and generically applicable indexing and retrieval techniques are available. The most popular indexing and retrieval techniques for classification-oriented systems are nearest neighbour matching and induction-based techniques like CART or ID_3. Nearest neighbour techniques do not require a class attribute and work well

on small sets of cases with many potentially relevant attributes. Machine learning-based techniques require a single class attribute to be defined for each case in the case base and work best on larger collections of cases where statistically relevant indexing rules can be created from an analysis of the historical cases. Each of these approaches has sound mathematical grounds and has been proven to be effective over many different kinds of classification problem domains.

- *Case adaptation needs are minimal (or non-existent) in classification systems.* Automatic case adaptation is important to the CBR developer who wants to deliver useful and easily maintainable applications. Since automatic case adaptation usually requires the collection and representation of knowledge that is fundamentally different from cases (*i.e.* rules, domain models, etc.), as well as the use of a different reasoning engine for the adaptation process, such mechanisms can be extremely hard and time consuming to build. Fortunately, classification-oriented systems are only concerned with determining how a new case should be categorised, not with how to adapt the similar case to solve the current problem. Because of this, classification-oriented systems avoid the pitfalls of creating and maintaining adaptation knowledge, making them easier to build and maintain. If automatic adaptation *is* required in a classification system, it is usually to perform a task such as the interpolation of numeric values of class attributes. This adaptation knowledge tends to be much easier to acquire than the type of adaptation knowledge required in planning/synthesis-oriented systems.
- *CBR tools are well equipped to handle classification-oriented tasks, making them easier to build.* Due primarily to the difficulty in creating generic mechanisms for performing automatic case adaptation, most commercial tools have focused on the problem of case representation, case indexing and case retrieval, where generic approaches do exist. This makes them well-suited to solving classification-oriented tasks.

In the remainder of this section we will cover various issues specific to different types of - but by no means *all* types of - classification system. As we will show, although each of the following problem domains are classification-oriented domains, they can differ significantly in their representation, indexing, retrieval and user interaction capabilities.

5.2.1 Diagnosis

Diagnosis is unequivocally a classification task. Given a set of symptoms, the system needs to determine the cause of the problem. Case-based diagnosis systems try to retrieve past cases whose symptom lists are similar in nature to that of the new case and suggest diagnoses based on the diagnoses in the best-matching retrieved cases. Current commercial CBR tools have been designed to solve diagnostic tasks and the majority of installed applications are of this type.

However, not all diagnostic tasks are the same. In some diagnostic domains, all the information is available at the beginning of the retrieval process. In this situation the system only needs to retrieve cases once, since no additional data will be collected that would change the best-matching cases. In other domains, the initial information about the problem may be sparse or incorrect. In these domains, the system performs case retrieval over several iterations or will query the user for information along the way to retrieving good cases. As more information is gathered, a diagnosis can be confidently presented to the user. Some domains have a well-defined set of diagnostic attributes and values for most

of them are available at run time. Other domains have very large attribute sets that are very sparsely populated at run time. Clearly, the latter situation presents difficulties in representation and indexing that must be addressed.

Help Desk

A help desk is a customer service area that specialises in handling problems with a product or service. Typically, a user calls for help in order to solve problems such as: *"Our printer is not printing correctly"*. The help desk operator must then try to understand why the customer is having the problem and suggest how it might be fixed. By providing the help desk operator with a case base of previously solved cases, the process of finding a solution is shortened and the level of expertise required is reduced. Finally, even if the help desk operator cannot solve the problem at once, he or she can at least identify an in-house specialist whose expertise can be called on. The use of help desk tools significantly reduces the number of calls made to the specialist on routine problems.

There are some important challenges associated with delivering a CBR help desk:

- *Handling the initial problem description as text*. Since the person calling in on the telephone is physically removed from the help desk operator and is typically uninitiated in the art of describing problems in a concise way, the initial problem description is usually a short, narrative description. Somehow, the CBR system needs to be able to find a starting point for diagnosis (*i.e.* retrieval) given this limited, free-form input. Several CBR tools allow indexing and retrieval based on case attributes that are essentially free text. Some tools break the text into small strings (n-grams) and carry out similarity-based matching of the string sets against historical cases broken up in the same way. Others build word-oriented diagrams that allow the developer to specify synonyms and stop-words. Either way, the goal is to find some initial set of historical cases that can be used as the starting point for more rigorous, attribute based retrieval.
- *Representing and indexing with sparse case attributes*. Because of the wide scope of a typical help desk problem, only a small proportion of the attributes of a case are likely to get values given them. The broader the type of problem that can be presented to the help desk, the sparser the case representation often becomes. This is due to the fact that different problem areas have fundamentally different attributes that need to be gathered to perform diagnosis in that area. Therefore, to build a general case base that covers every area, cases must have attributes for all of the possible features for every problem area, even if only a small fraction of the attribute set is ever filled out in a given case. The problem with sparse representations is that the traditional indexing and retrieval techniques (nearest neighbour and decision trees) do not work as well on sparsely populated cases. Some commercial tools have augmented their indexing approaches to deal with this problem. Another approach that can be used to avoid this problem is to break up the system into collections of problem-area specific cases. In each case collection, the representation of information is more densely filled. Using this approach requires the development of a top level case representation that can take the initial problem description information and do a good job of classifying the general problem area. Once this first classification is performed, the system passes the initial information into the appropriate collection of cases for specific diagnostic retrieval.

- *Gathering diagnostic information after the initial problem description*. In a typical help desk, the initial problem description is really just the starting point for performing diagnosis. The help desk operator must be able to ask the caller questions that will help him to narrow down the possibilities and finally arrive at a likely diagnosis. There are two common approaches to gathering additional information from the user. One is to ask the help desk operator (who then asks the caller) a sequence of questions, one at a time, that will eventually lead to a collection of relevant cases and a diagnosis. Another way is to present the help desk operator with a collection of cases given the information so far, and allow him or her to select which questions to ask the user based on an examination of the best cases so far. As more information is gathered, case retrieval is repeated until a case is found that is highly similar to the input situation. As it turns out, both of these approaches is good, but for different target users. Asking a sequence of questions tends to be a good approach when the help desk operators are novices. It allows them to get to a solution without necessarily knowing how or why they got there. For expert help desk operators, this approach tends to be too restrictive, forcing them to answer questions that they know to be irrelevant to the given problem based on their expertise. Experts would rather look at cases (even though they were retrieved with limited information) and pick and choose what further information to enter based on their knowledge of the likely causes for a particular problem. The second approach is thus more suited to expert operators.

- *Deciding between competing diagnoses*. A common problem for help desk operators, particularly for novices, is how to decide between competing diagnoses. Sometimes, the same set of symptoms leads to different diagnoses. In a CBR help desk system, a given set of retrieved cases might all be close matches but might nevertheless suggest different diagnoses. When possible, more information gathering can eliminate the confusion. Unfortunately, to simplify the case representation, many systems do not represent case information in enough detail to distinguish between subtle differences in symptomatic information that may lead to different diagnoses. The usual approach to resolving conflicts in diagnoses is to select the recovery action that is either the least costly to perform or the one that is most probable across the all cases with this symptom signature.

- *Determining which cases to add to the case base*. The same problem is often presented to help desks many times. Such situations are usually easily handled with very well understood prototypical cases that provide the common solutions to the common symptoms. This allows the CBR system to cover a large fraction of problem calls with a minimum of cases but does not suggest how to grow the case base over time. Intuitively, only new and interesting cases should be added to the historic case base. To decide which new cases to store in the case base many installed CBR systems ask the help desk operator to provide a "novelty" score for new cases, and also to rate them as to the usefulness of those cases that were retrieved as matches against the new case. If there were no good cases that addressed the problem at hand, the new case is a good candidate for inclusion in the case base (once the correct diagnosis for that case has been determined).

General Diagnosis and Fault Recovery

Help desks are not the only type of diagnostic systems that have been built using CBR. Case-based fault recovery and medical diagnosis systems have also been built. Because more symptomatic information can typically be collected in non- help desk diagnostic systems due to the fact that diagnosticians can directly check the object (*i.e.* the device is not on the other end of a phone line), case representations are often more sophisticated and the indexing mechanisms more robust. These are some of the issues in general diagnosis and fault recovery systems.

- *Diagnosis in domains with structured case representations*. Diagnosis in domains such as medicine require working with complex patient records. Patient records have important diagnostic information in them that cannot easily be represented using a flat attribute vector. For example, the patient record will have history information on the patient as well as chronological information on recent doctor visits or test results. This structured and chronological information is important in performing good diagnosis. In order to use these types of structured representations with existing CBR tools, developers are usually forced to aggregate the structured patient record into a flat attribute vector representation for indexing and retrieval purposes. Chronological information is summarised into single aggregate attributes such as averages, maximums, slopes, etc. Although the aggregation process seems to work well, it certainly results in the loss of potentially vital information.
- *Multi-phased diagnosis*. Occasionally, in order to simplify the case representation, a case-based diagnostic system will use a multi-phased approach to performing diagnosis. The first phase is to determine the general diagnostic category for the input symptoms. By eliminating the general diagnostic categories that are not appropriate for this case, the system can focus on a more detailed representation in the subsequent phases to retrieve cases with better overall accuracy. This approach simplifies the case representation and allows cases in a particular diagnostic category to be stored with their own unique representation and indexing mechanism.
- *Diagnosis and Recovery as two classification tasks*. Once a diagnosis is determined the diagnostician must then choose a course of action for fixing the problem. Determining the recovery action is also a classification task that can be addressed using the same cases that were used to perform the initial diagnosis. In fact, the diagnosis is used as an important indexing feature in determining the appropriate recovery action. Case-based fault recovery systems first determine the diagnosis using the symptoms to find relevant cases with a case index that is tailored to the diagnostic class attribute. Then they take the proposed diagnosis from the first step and look for recovery actions by retrieving cases (from the same case base) using a case index that is tailored to the recovery class attribute in the case.
- *Integrating other available domain knowledge into the CBR process*. Although cases are typically the best form of knowledge to use in a diagnostic system, they are by no means the only source of knowledge that is available in most situations. Often there are well-established rules and procedures for handling many of the most common sets of symptoms. There is no need to use cases if the rules are well-defined for certain common circumstances. In other, rarer instances, a model-based system can be used that can typically handle a larger set of

problems than collections of rules. In either event, a hybrid system can be built that uses rules or a model to handle the commonly seen problems, and cases for the remaining problems.

5.2.2 Prediction/Forecasting

Prediction and forecasting are common and important business problems. They can be regarded as classification tasks for the purposes of building CBR systems in these areas. A CBR approach to prediction or forecasting involves choosing a forecast number by looking for past cases that model the characteristics of the new case and using the observed value from the best historic case (*e.g.* stock value, inventory level, gross sales, etc.) as the forecast value for the new case. Case representation typically involves capturing a fairly stable set of attributes that are thought to be predictive of the observed class attribute. If enough cases can be gathered, machine learning techniques are the best choice for building accurate indexing mechanisms. Case-based approaches to forecasting and prediction have advantages over statistical models, particularly in the area of explaining their behaviour by presenting precedent cases of actual past events. This helps the users' confidence in the underlying decision making of the system. The following are common issues in building CBR systems for forecasting and prediction:

- *Handling temporal information in case representation and performing aggregation.* Predictions and forecasts are attempts to project future results by relating them to past history. By their very nature, forecasting and prediction tasks have a temporal element. For example, in stock prediction we would want to try to predict the rate of return of a particular stock based on the current state of the company and its history over the last several months or years depending on how far into the future the prediction is trying to go. For example, if you are trying to predict the rate of return over the next six months, it is probably not important to have information at the level of granularity of a day or maybe even a week. In order to use techniques like machine learning to derive good case indices, the representation needs to be "flattened" into an attribute vector. This usually requires aggregation over the temporal aspects of the historic information on which the forecast is based. For example, if you had monthly balance sheet or sales information on a particular company over the previous 12 months you would need to do certain things such as determine the slope of the trend line for sales or income in order to summarise the individual monthly information. Although aggregation is a valuable tool for building a suitable representation for prediction, it is also acceptable to use specific past data points as long as the number of historic data points in the final representation does not get too large. Based on the authors' experience, a good rule of thumb is to use a maximum of five historic data points. Any more than that and you would end up confusing, rather than enhancing, the system's ability to predict.
- *Dealing with large sets of attributes that are all relatively weak predictors of the class.* In the majority of domains in which we want to make predictions and forecasts there are many factors that we cannot account for that have a profound effect on the ultimate value we are trying to predict. As a result, we end up having to use case attributes that correlate with the class only very tenuously. Our ability to create an accurate case index is obviously not helped by this fact. In order to address this problem, a good developer should look for ways to augment the attribute set by creating composite attributes that are based on the original raw

attributes in the initial case representation. For example, rather than taking the net assets and net liabilities of a company and using them directly as indexing attributes, one idea is to take the ratio of these values: this could be a much better predictor than the individual components alone. Some CBR tools provide means of augmenting the attribute set of the initial case representation. Such features range from being simple to very sophisticated capabilities. Taking advantage of these capabilities almost always yields more accurate results.

- *Refining the forecast/prediction based on differences between the best case and the new case*. The cases that are returned from the initial retrieval are likely to differ in certain respects from the new case, even on the attributes that were used to retrieve the best matching cases. Because of this, the final forecast must be adjusted to take into account these differences. This is essentially a case adaptation problem. Whilst, in general, case adaptation knowledge is hard to come by and hard to codify and use, this type of "interpolative" adaptation is probably the easiest to carry out. Usually, adaptation rules are created that look at the differences between significant attributes of the new case and the best retrieved cases and suggest a small incremental change in the recommended forecast or prediction. Each difference either adds or subtracts a little from the forecast, usually based on a simple calculation, and the net result of all the incremental changes is the suggested (*i.e.* adapted) forecast. This type of adaptation is the most common in classification-oriented systems that have a numeric class attribute rather than an enumerated one (*i.e.* true or false, high, medium, low, etc.).

5.2.3 Assessment

Assessment tasks cover a wide range of business areas including: loan, credit and insurance risk determination; claim damage assessment, cost estimation and real estate appraisal, to name but a few. The goal is to assign a value (sometimes an enumerated value) to something by comparing it to the known value of something similar. In many ways this task is rather like the prediction and forecasting task. The main difference is that there is usually more relevant information to work with in assigning an accurate value. The representation is usually simpler as often there is no need to represent temporal information - as there is in prediction/forecasting tasks. We do not have to try to predict the future, but rather to evaluate the state of the present - which is a much easier task. Interpolative adaptation is also often required in carrying out case-based assessment. There are no particularly significant issues in creating case-based assessment systems that have not already been discussed. Assessment tasks are often well-suited to a CBR approach and existing commercial tools may be used to good effect in their development and delivery.

5.2.4 Adversarial Reasoning

Legal reasoning, dispute mediation and trade negotiation are examples of adversarial reasoning tasks. What distinguishes them from other classification tasks is that there are two competing sides, each trying to classify the new case in a way most suited to their different perspectives.

For example, in a legal dispute, the plaintiff's lawyer tries to find historic cases, whose details are similar to the current facts, which led to a verdict of "guilty". The defendant's lawyer is trying to find

cases similar to the same set of facts, but that led to a verdict of "not guilty". Obviously, only one side can win in Law, although a well-presented case might mitigate the damages against, or sentence given to, the defendant, if found guilty. In dispute mediation and trade negotiation, each side is trying to convince the other, or an arbitrator, that they should be getting more of the concessions based on other similar contracts or deals negotiated in the past. As in the legal case, the same set of base facts is used to try to support competing claims.

In adversarial reasoning, if you want to win, it is not enough just to support your own position based on the current facts. You need to be able to anticipate the arguments that will be made by the opponent to support their claim and refute them. For this reason, case-based systems for adversarial reasoning have a unique 'point-counterpoint' reasoning mechanism built into them. Despite this additional control mechanism, at the heart of the problem, classification is still the basic task. Some of the issues associated with case-based adversarial reasoning systems are:

- *Converting text into a more structured representation*. In most adversarial reasoning domains, cases are represented as freeform text narratives describing the situation, the arguments, the verdict and the reason for the verdict. Whilst this is appropriate for use by human experts, it is not appropriate for building a robust CBR system. The text needs to be converted into a more structured representation so that good indexes can be built for finding relevant past cases. Unfortunately, this task must currently be performed manually, making it a major bottleneck to building robust CBR systems in this domain. Recent advances in text retrieval and automatic text extraction technology may make this task much easier or eliminate it altogether.
- *Abstracting higher level attributes from the raw facts of the case*. Usually, the raw facts of the case are at too low a level of granularity to be useful in finding relevant cases. For example, if an assault was made with a large piece of glass it would be unnecessarily restrictive to look for other assaults that were also made with a piece of glass. A better approach would be to abstract the raw attribute into a new attribute such as: "Was the weapon a deadly weapon?" This enables the system to search for relevant historic cases with the generally more useful abstract attribute rather than the initial raw fact. As previously mentioned, commercial CBR tools provide various mechanisms for abstracting raw case attributes into new ones. The key to good attributes abstraction is careful knowledge engineering.
- *Building the point/counterpoint mechanism*. When cases are retrieved from the case base based on the abstracted attributes of the new case, they are likely to include cases with different classes - both desirable and undesirable outcomes. The initial hope is that a high percentage of the retrieved cases will be in the class required. If not, the few cases in the class required have to examined to find the significant specific similarities between them and the facts of the current case. The aim is to find similarities along the dimension that is furthest away from the cases with the undesirable classes. This 'dimension' then becomes a distinguishing factor and can then be used to support the arguments even in the face of a preponderance of case evidence against the position. In addition to looking for key similarities between the facts and the cases that support the hoped for outcome, the indexing structure itself can be used to help find factors that, if they could be changed or minimised, would lead

to a much more favourable set of cases. This type of 'what if' reasoning can be performed if the cases are indexed using a hierarchical index, built either by hand or with a machine learning technique. Essentially, the point/counterpoint mechanism attempts to emphasise factual links to 'good' historic cases and de-emphasise factual links to 'bad' historic cases through attribute based analysis and 'what-if' case retrieval.

- *Integrating other types of reasoning.* In many adversarial domains, particularly the law, there are statutes in addition to cases that must be taken into account to determine guilt or innocence. These statutes can typically be thought of, and represented as, rules. In order to take into account this rule-like information, adversarial reasoning CBR systems need to have a mechanism that supports rule-based reasoning. One of the difficult aspects of using statutory rules in adversarial reasoning is that the wording of the statutes often leaves a lot of leeway when interpreting whether or not a given statute applies to a given situation. The preconditions for the rules are not grounded in hard facts, but rather in abstract concepts such as 'reasonable care'. In such situations we can use CBR as a mechanism for determining whether a particular abstract precondition is true or not. This is also a classification task.

5.2.5 Classification Tasks - Miscellaneous Issues

The previous sections on different classification tasks illustrated the difference in the nature of the classification when moving from one domain to the next. Despite the differences, all these domains are linked by some common factors. We will close the discussion of classification tasks by looking at some of these factors and the issues that must be addressed when building case-based classification systems.

- To accommodate current tools, the case representation must end up in a flattened form (except for KATE and S³-CASE). This sometimes requires the use of aggregation to summarise a structured representation into a flat set of attributes.
- Text handling is rudimentary and should not be heavily relied upon for indexing cases. If text is used it should be no more than one or two sentences' worth of information per field.
- Current machine learning based indexing approaches can only handle a single class attribute at a time. Either create a single class attribute as a composite of several raw attributes or use nearest neighbour matching.
- If enough cases can be collected, machine learning techniques are best for performing classification relative to nearest neighbour techniques. However, if cases are not "clean" *i.e.* - that is, free from noise and errors - CBR can perform better than standard techniques.
- Augmenting the raw attribute set is usually necessary and almost always a good idea. Regardless of which indexing approach is used, getting the right attribute set is much more important. Take advantage of the commercial tools' capabilities to augment the attribute set and do not be afraid to experiment.
- Case adaptation in classification systems is usually restricted to interpolation of numeric classes. It is a good idea, where possible, to 'discretise' numeric class attributes into ranges. This improves the system's accuracy and ability to make good indexes and reduces the need to perform adaptation in certain situations. If adaptation is required, the best approach is to build an adaptation mechanism that uses simple rules that incrementally alter the final value.

- New cases should be added when the system makes a bad choice rather than when it makes a good one. Whilst this may seem counter-intuitive at first, it is in fact the best approach. When the system makes a mistake it is showing a lack of case coverage in that area of the problem domain. By adding cases that were not correctly classified we can strengthen these weak regions of the problem space without overfilling the case base with cases and reducing overall retrieval time.

5.3 Planning / Synthesis Tasks

Planning and synthesis tasks are also ubiquitous in business and industry but have so far been very difficult to automate in a robust fashion, regardless of the technique being used (*i.e.* rule-based, model-based, case-based, etc.). Generally, it is much more difficult to create an artefact from a set of specifications than it is to classify an existing artefact against a set of known prototypes. Case-based planning/synthesis systems seek to simplify the task of creating a (potentially abstract) artefact from a set of specifications by using past cases to 'bootstrap' the process. This bootstrapping allows the planning/synthesis process to start from some "known-to-be-good" previous plan or design, rather than starting from scratch. The leverage in this approach stems from the fact that modifying a good initial plan/design is much easier than creating a new one. Whilst academic research results are showing that this is probably true, it is far from easy to accomplished in practice. Because of this, there are few installed systems (though many research systems) that go beyond the initial retrieval of candidate cases that a human user can then use and modify. There are several reasons why it is difficult to build planning/synthesis systems:

- *The case representation of a plan, design or schedule tends to be a complex, structured representation.* Plans, designs and schedules are complex artefacts. Representing cases for plans or schedules requires representing temporal concepts and relationships. Designs and configurations require the representation of spatial or other logical relationships between various sub-components of the design or configuration. This results in a highly structured and complex case representation. Whilst it is often possible to define such complex case representations, they tend to be difficult to work with (*i.e.* index, retrieve, adapt, maintain, etc.). Commercial tools are ill equipped to handle these types of complex representations, so the developer is usually required to build such representations from scratch.
- *Because of the complex representation, fewer cases are collected.* Because the representation designed for problem solving of this type is usually radically different from the way information about the artefacts is commonly stored (presuming that they are stored electronically), cases must be entered manually - usually by experts who have neither the time nor the desire to do so. This is a major impediment to building case bases of a size sufficient to provide good coverage. Because so few cases are usually collected, case adaptation becomes even more critical in these types of system.
- *There are few generic indexing/retrieval techniques available for structured representations - most are home-grown and ad-hoc.* Structured case representations, although well suited to describing planning and synthesis tasks, do not have well defined or generic approaches for indexing or retrieving. As a result, the indexing strategies used are either very ad-hoc and problem specific, or limit the indexing/retrieval task to a subset of the case attributes

represented using a flat attribute vector. Although this simplification allows for retrieval of cases, retrieval accuracy suffers because the full context and structure of the historical cases are not used to find the best matches. Consequently, the system is again forced to rely more heavily on adaptation mechanisms to overcome potentially poor initial case retrieval.

- *Case adaptation is usually a major part of creating a planning/synthesis system, and adaptation is difficult in planning/synthesis tasks.* As the previous points have suggested, case adaptation becomes a critical component in building planning/synthesis systems even if enough cases are gathered and an adequate indexing/retrieval scheme is developed. In a classification system it is enough to know that a similar case has a particular label (class), even if it is not identical to the new case. Planning/ synthesis systems must be able to satisfy all of the specifications of a given new case even if a very close-matching case is found. This requires knowledge. Unfortunately, the knowledge required to adapt cases tends to be at a lower level of granularity than the cases themselves. Adaptation knowledge is usually in the form of rules. These rules are used both to identify the salient differences between the input specification (new case) and the best-matching case and to suggest modifications to the best matching case that will satisfy all of the input specifications for the artefact to be created. For many planning/synthesis domains, this knowledge is very hard to come by. Even if it can be codified, adaptation then requires the incorporation of alternate reasoning paradigms into the case-based architecture. Building and maintaining such systems (particularly from scratch) requires expertise that many potential developers of such systems do not have.
- *Problem solving must be sophisticated enough in an installed application to be useful to experts doing real work.* An installed application must be of value to its users in carrying out their daily tasks. They will not be impressed with the level of effort that went into creating an automated adaptation mechanism if that mechanism is not robust enough to help them find solutions to real problems. Particularly in planning/synthesis tasks, the end users are highly educated and take pride and pleasure in their ability to come up with creative solutions to complex problems. Automated planning/design aids almost always fall far short of users' expectations, particularly when offering end-to-end capabilities (*i.e.* no human in the loop). As a result, the few successful installed applications in this area focus very heavily on retrieving good starting cases and leaving adaptation to the human experts.
- *Commercial CBR tools do not address planning/synthesis tasks very well beyond initial plan/design case retrieval, if at all.* Commercial CBR tools to date fail to provide robust development capabilities for building planning/synthesis task problem solvers. As regards representation, indexing, retrieval and adaptation, creating planning/synthesis problem solvers is of a higher order of magnitude of difficulty than building classification-oriented problem solvers. There are no satisfactory, generic approaches to building these systems. This is the major impediment to CBR tool developers properly addressing planning/synthesis tasks beyond providing some sort of rule based inferencing capability within their existing case representation/retrieval frameworks. Further research is needed in developing useful knowledge engineering methodologies and generic representation and indexing approaches that work on structured cases before we will see good CBR tools that can adequately attack a broad range of planning/synthesis tasks.

The remainder of this section will examine some specific planning/synthesis tasks and the challenge of using CBR approaches in solving these tasks. Although the preceding argument casts a bleak light on

the use of CBR systems for planning and synthesis tasks, there are ways to make good use of historic cases and avoid some of the aforementioned problems. These approaches will also be addressed below.

5.3.1 Planning

A plan is simply a sequence of steps that will accomplish some goal starting from some initial state. Planning has long been a topic of interest and research in Artificial Intelligence. Traditional planners use sophisticated, state-space search mechanisms that work on low level operators (*i.e.* components that make up a plan). These systems synthesise a plan to accomplish some goal from an initial state by piecing the plan together, bit by bit. Because the search process can be intractable for even moderately complex domains, this type of completely bottom-up approach does not work well in the real world. Over time, the developers of these planners realised that human planners do not usually start with a clean slate when given a new planning task. Instead they work from some previous plan and then make small changes to bring it in line with the new objectives. This is essentially a case-based approach to planning: past plans are stored as cases in a case memory and are retrieved relative to the description of the initial state and the goal.

One of the main factors in building a case-based planner is in deciding how large the cases should be. Should a case be a full plan, a logical chunk of a plan or a small step of the plan ? The choice of granularity will have a profound effect on the issues one must address in building a case-based planner.

Cases as Large Plan Chunks

When large plan chunks are stored as cases the representation tends to be more complex in order to accommodate all the various steps in the plan in a single case. This makes indexing the cases harder due to the lack of good indexing/retrieval techniques for handling highly structured cases. Case adaptation becomes a very important task because there tend to be fewer cases in the case base (since they are large chunks) and no single case has all the right elements of the plan that are needed for the given situation. Adaptation then becomes a more difficult task because the adaptation knowledge must know how to handle changes in the context of a much larger plan that has more elements that potentially need to be adapted. On the positive side, if a good case is found, the amount of work that needs to be carried out to adapt the plan can be much less than if the plan had to be put together from smaller chunks. In general, if the goal is to provide a mechanism for aiding a human user in building a good plan from some reasonable starting point - and automatic adaptation can be avoided - this approach is good. Otherwise, building the automatic adaptation mechanism will overwhelm the process of building the system and jeopardise its ultimate effectiveness.

Cases as Small Plan Chunks

If the goal is to build a planner that has to construct a completely viable plan, or that must be able to alter the plan as it is being run, it is often better to use much smaller plan chunks as cases rather than larger chunks. By using smaller plan chunks, the system can construct a viable plan by essentially doing a case-based, state-space search. Small chunks tend to be easier to represent and therefore easier to index in useful ways. By using case chunks that are larger than the atomic operators, but logically complete in handling some sub-goal, we can use a traditional plan generation approach without having

an intractable search problem. Obviously, using this type of approach requires the construction of a control structure that will perform the state-space search using case retrieval to generate viable states. Commercial CBR tools do not provide this type of capability. Once the control mechanism is built it can be used not only to generate plans, but also to re-plan if unforeseen events prevent certain pre-planned steps from properly executing. By using small chunks as cases and having a simple control mechanism, we can avoid the adaptation difficulties that plague the large chunk option. On the negative side, the small plan chunks are not really useful if the control mechanism is not built, eliminating the possibility of providing 'quick and dirty' solutions by retrieving large plan chunks (even without adaptation).

5.3.2 Design

Design is perhaps one of the most difficult tasks that is performed in business and industry. The designer is asked to create an artefact that does not yet exist, from component parts that do, in conformance to an initial specification as to the appearance and function of the artefact. To make matters worse, the specification usually states conflicting requirements. For example, a plane is required that is as light as possible, as strong as possible and as inexpensive to build as possible. Unfortunately, maximising any one of these requirements is likely to affect the other two negatively. We could make a very light plane out of balsa wood but it would not be strong enough to hold passengers; we could use composite materials to make it both light and strong but it would be very expensive; we could use a cheap, strong material like aluminium, but it might not be light enough to suit our needs, etc. The designer must be able to balance these conflicting goals.

Design has been a popular target for AI research primarily due to the allure of the challenge, but also due to the potential payoff of a system that can automatically create a design from a set of specifications with little or no human intervention. Whether it is software, circuit, mechanical, architectural or menu design, the basic task is the same. It is only the knowledge, used to create the artefact, that is different. As with the case of planning, human experts usually start from an existing design that already has some of the basic characteristics that are called for in the specification. The initial design is then adjusted to meet the specification. This makes design a suitable domain for CBR systems if a few pitfalls can be avoided.

Case-based design has the same issues concerning the size of the cases the case base as does case-based planning. In addition, design often requires a case representation that must handle the spatial or logical aspects of design. As is the case with any complex representation, indexing and retrieval becomes difficult and often ad-hoc. Case adaptation in design is usually even more difficult than in planning because there are many possible ways to arrive at a suitable design that meets the specification, depending on which portions of the design specification are optimised. In addition, the experts do not always know in advance what effect a design change will have on the final design, particularly in the context of other parts of the design. This makes it difficult to accumulate good design rules for use in adaptation. This may be why circuit design is a more popular target than software or mechanical design for AI systems, given the more rigid and predictable effects of making changes to a circuit.

The chances of success are much greater if a case-based design system acts as an assistant to a human designer (by only retrieving historic cases), than if it tries to replace the human designer (by trying to carry out case adaptation as well). Commercial CBR tools can support the retrieval step, but they are

not well suited to perform the adaptation of design cases. The knowledge that would be required to perform useful, robust adaptation on real world problems depends on the domain and must be programmed by hand using classical technologies such as C, a Hypertext language (for example, Asymetrix *ToolBook's* Open Script language), or a rule-based language (such as Inference's ART-Enterprise's objects and rule language).

5.3.3 Configuration

Configuration is a special case of design. In a configuration task we need to put together a set of known components in a way that is compatible with a certain specification. In computer configuration, we need to select and hook together a limited set of components in a way that meets the user's needs for processing, connectivity, storage and upgradability. In autoclave configuration (see Chapter 6 on *CLAVIER*), we need to spatially arrange a set of parts inside an autoclave so that they all cure satisfactorily. Case-based configuration allows us to avoid having to create a configuration from scratch by starting with a historic case that can be modified if need be. What makes configuration easier than design and more likely to result in a successful application is that:

- User requirements for configuration tend to repeat themselves, making it more likely that there will be just the right case in memory. In fact, for many domains there are a set of predefined template configurations that are used in most situations and modified when necessary.
- Because configuration is often a spatial reasoning task (something humans are much better at than computers), graphical editors can be built that allow the human expert to make the modifications to the best-retrieved configuration, avoiding the need for automatic adaptation mechanisms.

Chapter 6: Clavier: A Sample Application

6.1 Optimal Configuration of Autoclave Loads

Over the past decade, aerospace manufacturers such Lockheed Missiles and Space Company have been using more and more "composite" materials rather than metals in the construction of aircraft, spacecraft and missile parts. Composite parts are made from layers of a graphite-epoxy 'fabric' that are formed into a single sub-structure through a curing process that takes place in a large, pressurised oven-like device called an "autoclave". (Some autoclaves can be as large as 20 feet in diameter and 50 feet in length). The resulting composite part has the desirable properties of being lightweight and very strong, relative to equivalent parts made from metals such as aluminium.

However, these desirable properties do not come cheaply. Individual composite parts for aircraft or missiles can cost tens of thousands of dollars, just for the materials. If the manufactured composite part is defective it cannot be melted down and recast; it can only be thrown away. Although there are many things that can affect the final quality of the composite part, the autoclaving process is essentially the point of no return - there is no recovery procedure that can save a poorly cured part.

The key to successfully curing a composite part lies in subjecting it to a fairly precise schedule of temperature and pressure change over a 6-7 hour period in the autoclave. Thermocouples attached to the part give the autoclave operator continuous information on whether the part is following the prescribed pattern. The operator can make small changes to the temperature and pressure in the autoclave to ensure that the part stays within tolerance levels and that it will emerge successfully from the process. This task is precarious even if there is just a single part in the autoclave. Unfortunately, manufacturing schedules require maximising the use of the autoclave over each cure cycle by curing as many parts as possible. In this situation, the operator must get *all* of the parts in a given load to adhere to the rigid temperature and pressure schedule that will ensure that each part is successfully cured.

This complex process is made even more difficult due to the fact that a given autoclave has different temperature characteristics in different locations inside it. Temperatures do not change consistently or instantaneously over all parts of the autoclave during the temperature adjusting process. Furthermore, a given load of parts in an autoclave run can be made up of both large and small parts, each with its own required signature for heating up and cooling down during the curing process. The goal is for all the parts in a given autoclave load to heat up (based on thermocouple readings) at the same rate within a relatively small tolerance level. Essentially, the difference in temperature between the "leader" (the hottest part in the load) and the "lagger"(the coolest part in the load) must be no larger than 30° F at any point during the 6-7 hour curing process (see Figure 6.1).

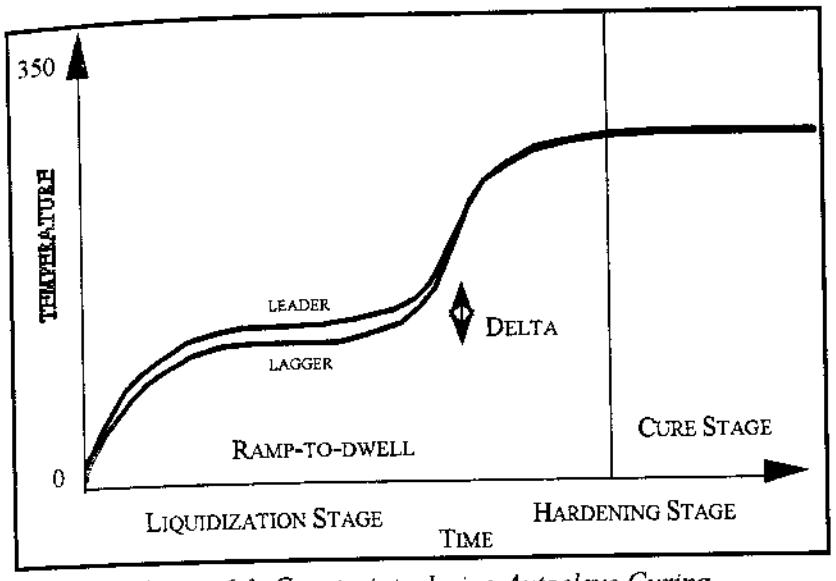

Figure 6.1. Constraints during Autoclave Curing

Clearly, coming up with the right combination the first time is vital. The most important step in this critical task is in correctly selecting and placing parts relative to each other inside the autoclave. This task is called autoclave configuration. "Compatible" parts are selected for a given autoclave load based on their individual heating characteristics and are strategically placed in the autoclave in positions that are likely to produce a consistent ramp up of temperature relative to all the other parts. Large parts that are slower to heat up are placed in the hotter parts of the oven and smaller parts tend to be put in the cooler parts of the oven (see Figure 6.2). Correctly selecting and placing parts in the oven so that they all heat up at the same rate is much more of an art than a science, and one which is based almost exclusively on 'evolutionary' trial and error over many runs.

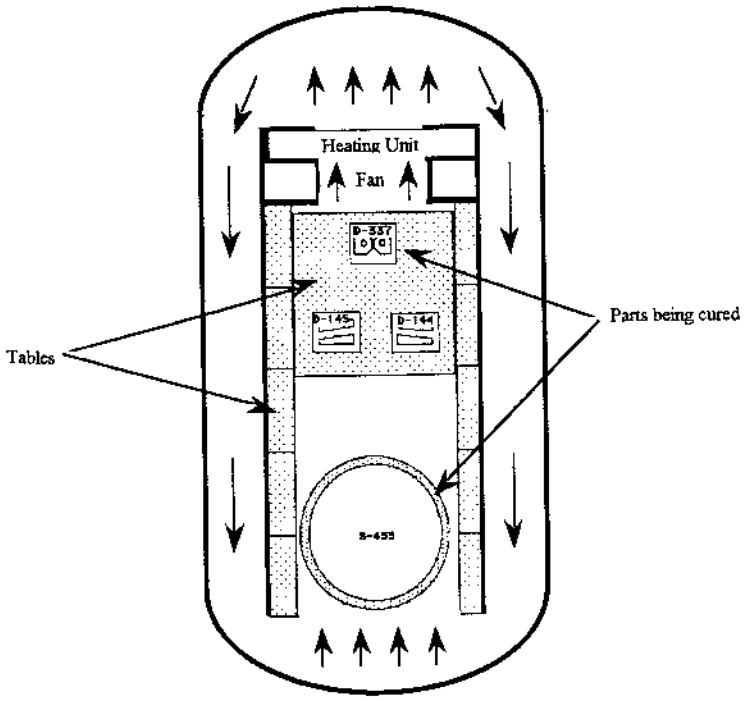

Figure 6.2. Autoclave Load

Prior to the development of *Clavier*, expert autoclave loaders at Lockheed kept track of their knowledge by keeping a notebook of successful configurations (*i.e.* drawings depicting the parts and layout of a load). Then, depending on which parts needed to be cured at any given time in the production schedule, the shop floor expert would manually select the historical configuration that: 1) was known to be good and, 2) would cure the greatest number of 'high priority' parts on his list. Occasionally, a good historic configuration cannot be completely filled out from the current list of required parts. The historic configuration might then be modified by either leaving those missing parts out of the current run or substituting parts with similar characteristics to the missing part. If a modified configuration was successfully run (*i.e.* all parts were cured properly), it would be noted in the notebook for future runs.

6.2 Clavier: A CBR System for Autoclave Configuration

6.2.1 Why CBR?

The researchers at the Lockheed AI Centre found this problem to be well-suited to a CBR approach for many reasons. A previous attempt to tackle this problem with a rule-based approach failed because the shop floor experts had an extremely difficult time trying to describe the rules for creating a good autoclave configuration in a way that could be easily represented and executed in a rule-based system. The rules were not very well understood, even by the experts. This was compounded by the difficult spatial reasoning that would be needed to synthesise a configuration from scratch.

The experts had thought in terms of cases and had archived their successes in such a form. They had used past cases as a starting point for the exploration and fine tuning of autoclave loads. Clearly, CBR would naturally be the most suitable technology for providing a problem-solving system that would be based in terms the experts could appreciate and have confidence in. Particularly when one considers that thousands of dollars were at stake in each autoclave load, the users needed to have complete confidence in the reasoning of the system. Providing precedent cases of past successful loads was the best way to convince the user of the validity of the CBR system's reasoning and to give them the confidence to use it in their daily routine.

6.2.2 Case Representation

The primary indexing features for cases in *Clavier* were the names of the parts in previous, successful runs of the autoclave. In addition to the part information, there was information about the table on which the parts were located, where the part was located on the table relative to the other parts and where that table was situated in the autoclave (in the front, middle, or back). This layout information was sufficient for performing effective retrieval of cases.

6.2.3 Case Indexing/Retrieval

The autoclave operator has a current list of the parts that are ready (or will be ready) to be cured in the autoclave over the course of a given week. Parts on the list are prioritised based on current assembly line requirements. Each load must be planned in advance and the operator needs to be aware of the priority of the parts on the list in configuring each load. For any given list of prioritised parts, *Clavier*'s

retrieval mechanism seeks to find the successful past loads that cured the largest number of high priority parts on the current list.

The retrieval mechanism is an associative matching scheme that tries to fill parts in the cases in the case base from highest to lowest priority (Figure 6.3). As a prioritised part is filled into a previous case, that case's score increases proportionally to the priority level (WIP = 3, High = 2, Low = 1). The highest scoring historic cases are then returned to the user for approval and, possibly, adaptation if the highest scoring case is not completely filled out with parts from the current parts list.

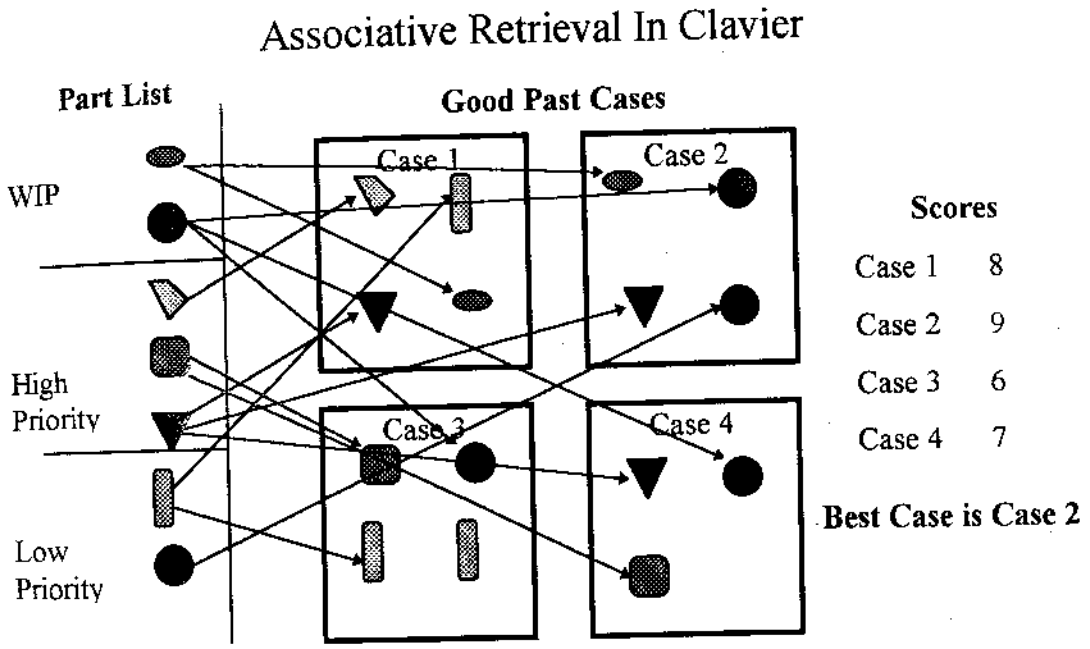

Figure 6.3. Associative Retrieval in Clavier

6.2.4 Case Adaptation

Case adaptation is considered when the best matching case is not completely filled out by parts from the parts list. An early version of *Clavier* attempted to provide an automatic mechanism for performing case adaptation. This automatic adaptation mechanism looked for substitute parts in other known layouts that were similar to the best retrieved layout. The idea was that if a similar part was in a similar place in a similar layout, it should safely be substitutable. Unfortunately this line of reasoning, although intuitively valid, was not found to be sufficiently compelling to the shop floor personnel. They did not want the system to suggest changes to a known successful case. They wanted to do that task themselves. They were particularly concerned by the fact that more novice autoclave operators might take the advice of a computer system's adaptation of the case and ruin the load.

Therefore, instead of performing automatic adaptation for the user, *Clavier* provides a graphical case "editor" that allows the user to adapt cases. In terms of programming effort and knowledge engineering, building an editor for user adaptation is much less costly than building an automatic adaptation mechanism. It also led to greater user acceptance of the system's advice. Once a case is edited, *Clavier* can provide validation guidance to the autoclave expert by showing past autoclave loads in the case

base that are similar to the adapted configuration. If these similar past cases turned out bad, *Clavier* would recommend not making the suggested adaptation. After the adapted case is actually run through the autoclave it is then reinserted into the case base for future use either in configuration (if it was successful) or validation (if it was unsuccessful). The overall architecture for *Clavier* is shown in Figure 6.4 below.

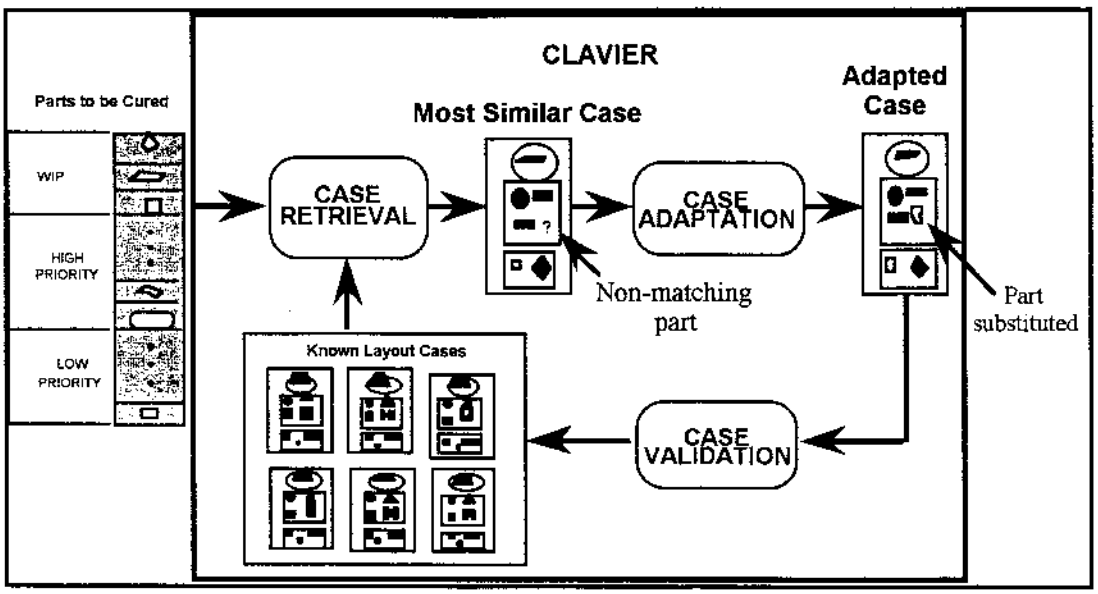

Figure 6.4. Overall Clavier Architecture

6.2.5 Scheduling Extensions to Clavier

The basic architecture shown in Figure 6.4 defines the process of configuring a single load of the autoclave from a prioritised parts list. As was previously stated, the autoclave operator needs to plan multiple loads in order to finish the entire list of parts according to a schedule. A natural extension to *Clavier* was the development of a multi-load scheduler.

Clavier's scheduler performs a state-space search using the basic single load configuration tool to generate states. The goal of the scheduler is to find the smallest collection of loads that will complete all of the parts on the parts list. The scheduler also takes into account part priority as well as the availability of manufacturing resources to get the parts ready in time for curing in a scheduled load.

6.2.6 System Issues and Effort

Clavier was written from scratch in Common Lisp for a Macintosh computer. The initial version was delivered in September 1990 after about 9 person-months of programming effort. This first version had the ability to configure individual loads and provided a case editor for user adaptation of cases. Version 2 of *Clavier* was delivered in November 1991, after another 15 person-months of programming effort. Version 2 expanded on Version 1 by providing load validation and multi-load planning, as well as various administrative and report generation capabilities.

6.3 The Use and Impact of Clavier on the Shop Floor

As one of the first installed applications of CBR, developed prior to the emergence of the many commercial CBR development tools, *Clavier* has been a tremendous success. The following is an excerpt from a recent article by David Hinkle and Christopher Toomey of the Lockheed AI Centre (Hinkle & Toomey, 1994), describing the net benefits of *Clavier* to Lockheed's composite manufacturing operations:

> "*Clavier* has been in continuous daily use at Lockheed's Composites Fabrication facility in Sunnyvale, California since September 1990. Two to three autoclave loads are cured per day in this facility, all of which are selected through operator consultations with *Clavier*. *Clavier* also generates hard copy reports of the autoclave loads that are used for record-keeping purposes. The system has recently been expanded for use in other Lockheed manufacturing facilities, and negotiations are under way for licensing the software to other aerospace companies...
>
> ... There are now five operators and two support personnel who regularly use the system as part of their daily routine for the generation of autoclave load configurations and other reports.
>
> If a mould goes outside the correct thermodynamic profile, a discrepancy report is issued and the part must be manually inspected at a cost of $1000. If the part is flawed and must be scrapped, it costs an average of $2000, but in some cases can cost between $20,000 - $50,000 for some parts! Since *Clavier* came on line, discrepancy reports due to incompatible loads have been virtually eliminated, saving thousands of dollars each month..."

Chapter 7: Evaluation Summary and Overview of the Tools

Please see the disclaimer in the section "About this Document" at the beginning of this Report.

This chapter summarises the essence of the report. For each tool we summarise its main characteristics and highlight the results of its evaluation. In the last sub-section we also provide an overall summary of the evaluation. The detailed evaluation is presented in Chapters 9 and 10.

We should view this evaluation as being part of a chain of actions that enhance the role of CBR within the field of information technology. We hope it contributes to identifying important application and research issues. We also hope it enables a more accurate assessment of what can be expected from CBR technology in the near future, with respect to the goal of being able to build software systems that are able to learn from their own problem solving experience[1].

7.1 CBR EXPRESS

Primarily designed for help desk applications, CBR EXPRESS appeared to be a tool that provides a number of standard CBR features combined with a comfortable user interface and surprisingly fast retrieval speed. The results of all tests were satisfying. With regard to some of the points mentioned later, CBR EXPRESS is very stable. Compared with some of the other tools in the evaluation, the expressiveness of the similarity measure could be better. CBR EXPRESS works well in domains that can be represented by flat attribute-value-vectors. If domain modelling requires background knowledge such as rules, formulas, constraints or taxonomies, CBR EXPRESS is not the best choice. However, this disadvantage disappears if the tool is used as a component of another application (*e.g.* ART Enterprise) instead of using it as a stand-alone application.

CBR EXPRESS was the only tool in this evaluation specifically designed to cope with the needs of an operator at a help desk. It includes a so-called "tracking panel" that allows the user to create customer records, access customer data and keep track of pending calls. Customer data can be held in an RDM or SQL database. The tracking and the search panelsare integrated to support an easy flow of information.

If a new case needs to be handled, the operator enters customer data (or searches for the data if the customer details have already been stored). If he is not experienced enough to handle the enquiry, he simply switches to the search panel to search the case base. Pressing a button brings him back to the tracking panel, where the retrieved solution is displayed and can be stored along with the customer data.

As a special feature, CBR EXPRESS allows a report generated from the current tracking base to be printed. Further support for the operator comes from the ability to browse textual or graphical

---

1 That the current state of technology is a long way from being able to achieve this can be seen, for example, by the answers to the last question in Table 9.26.

explanations for each case or question, if available. This on-line documentation could be, for example, a technical drawing or a repair description.

We now summarise some special characteristics of the tool:

- As shown by test T3, **CBR EXPRESS** allows cases to be retrieved in a very short time (less than 1.5 seconds for 1,000 cases). This time does not appear to increase linearly with the number of cases. This becomes important for case bases with some thousands of cases where lack of an inductive component may become a problem.
- One reason for the fast retrieval is the use of a textual description to narrow down the number of cases to be considered. As a first step in each search, the user normally enters a textual description of the new case. **CBR EXPRESS** identifies the keywords, transforms the description and matches it against the case base. This works quite well if the description is carefully chosen.
- Unlike other tools, the target attribute of a case is not just an attribute but rather, more general information. It may be seen as a data record that associates additional information with the case. As mentioned before, **CBR EXPRESS** allows graphics browsing of action procedures (via external *ToolBook* pages). Since those are defined separately from cases and questions, the same action can be shared by different cases.
- **CBR EXPRESS** distinguishes between user and maintenance mode. Only the maintenance mode allows changes of the similarity measure, entering questions or reindexing the case base. In user mode, most of the windows are removed to prevent the user from destroying the case base. The menus are reduced to only those features necessary for retrieval.
- The user may modify the *ToolBook* interface to **CBR EXPRESS** by adding his own objects to satisfy special needs.
- **CBR EXPRESS** operates in multi-user mode on any network that supports the IBM NetBIOS standard (*e.g.* Novell 3.01). Multiple users can therefore share the same database.

Finally, we would like to mention some details that should be kept in mind when building an application.

- It is worth reading the manual carefully about the question (attribute) weights. To quote from the manual: "The case base author is advised to leave these weights at their default settings [...]. The default settings have been selected to produce the most intuitive behaviour for the comfort of new users". We did change the weights and found that numerical attributes are simply not considered if their match weight is increased to 40 or more. To avoid this problem, the weight of numerical attributes should initially be set to lower numbers. The other weights can be set with regard to these values later.
- Although the textual description helps to guide the search, it can be a handicap if there simply are no descriptions (as would probably be the case, for example, if the cases were generated automatically). **CBR EXPRESS** would not run in user-mode without these descriptions. Even dummies would not help.
- Because questions are taken from currently retrieved cases during iterative search, domains with many unknown values (such as the CNC MACHINE TOOL domain) cause problems. After entering the description, the questions from the five best matching cases are presented to the user. It may be that important questions are left out because they are set to "unknown" in each of the five cases. The user has no opportunity to answer any such omitted questions.

7.2 ESTEEM

ESTEEM's main advantages are the possibility of using nested case structures and the rule mechanism. Both of these features have been inherited from the underlying Kappa-PC, an object-oriented **development system** for rule-based expert systems in which ESTEEM has been built. Rules allow ESTEEM to adapt cases automatically and to compute similarity measures and weights during the retrieval process. Owners of a full version of Kappa-PC (the version delivered with ESTEEM is only a runtime library) are able to program even more features themselves. However, as rules modify ESTEEM's internal data directly, they can only be written and understood by experienced programmers.

A problem with the user interface, that does not affect ESTEEM's functionality, is that some parts of it, such as scroll bars, do not behave as one might expect in a Windows application. In particular, some of them are remarkably slow.

In general, ESTEEM offers a nearest neighbour matching algorithm with some powerful extra features and a programmable adaptation mechanism provided by Kappa-PC's rules. Because of the disadvantages mentioned above and the lack of an explicit mechanism for handling missing values, efficient work with ESTEEM is only possible on case bases with just a few missing or unknown values. Despite these restrictions however, ESTEEM can handle a large number of possible applications. An example of a successful application is support in technical troubleshooting that can be used by non-technical people. Other possible applications, such as simple medical diagnosis, help desk applications and financial applications are provided as sample case bases delivered with ESTEEM.

7.3 KATE

The KATE toolbox consists of, amongst others things, KATE-INDUCTION and KATE-CBR. Both components can handle complex data represented by structured objects, relations and general knowledge about the domain. KATE-INDUCTION automatically builds a decision tree from a database of training cases. At each node in the decision tree, it evaluates the information gain for all the attributes that are relevant and picks the one that yields the highest increase of the information gain measure. At each node however, KATE-INDUCTION performs additional work to compute the set of candidates - that is, the attributes of objects - that are relevant in the current context. Information theory guarantees the correctness and the effectiveness of the resulting tree. Building the tree is fast and efficient (see Table 9.24a). However, an induction tree cannot handle unknown values in an optimal way during consultation. A decision tree is not incremental and the consultation system is static. Once built, a tree will not automatically be changed afterwards. KATE-INDUCTION does not differentiate between different kinds of user. The consultation of the tree is the same for all kinds of user and one always has to use an entire path of the tree to reach a conclusion. One can summarise the requirements for using the KATE-INDUCTION component as follows: the final system has to quickly deliver a diagnosis after asking a small number of questions; the end user has no ability to modify the system behaviour and follows exactly the predefined set of questions; the environment does not evolve very much over time.

Instead of using a generalisation of a database as in the KATE-INDUCTION component, KATE-CBR dynamically builds a path leading to the most similar cases, with respect to the wishes of the user. Thus, this toolbox component allows more consultation flexibility. The test selection component uses

the same information gain function as the induction component, but adapts it dynamically during the consultation phase. Unlike KATE-INDUCTION, KATE-CBR may be adapted for different users. In addition, KATE-CBR is better at handling incomplete and noisy data in the consultation mode because it does not perform generalisations.

The main disadvantage of using the version of the toolbox we evaluated was in building the system from scratch. We had a description in ***Casuel*** syntax for the domains used in the tests. It was therefore easy to build up the development system because the KATE toolbox has an interface to parse a domain definition in *Casuel* syntax or in *Excel* format. However, if a *new* domain description has to be designed, the only way to build up a development system in the version we evaluated was to construct a definition in *Casuel* syntax, or by using *Excel* on a database. It is therefore necessary to be familiar with the syntax of these languages, and in complex descriptions it is easy to lose the overview of the definitions. A descriptive model editor and questionnaire are incorporated in KATE-EDITOR, but we did not evaluate it.

The main advantage of the KATE toolbox is that it offers two complementary techniques that can be used with the same development system description. A second advantage is its ability to handle complex object-oriented domains.

7.4 REMIND

The key feature of REMIND is its ability to use background knowledge from experts to improve indexing, retrieval and similarity assessment. Figure 7.1 summarises the different kinds of knowledge available.

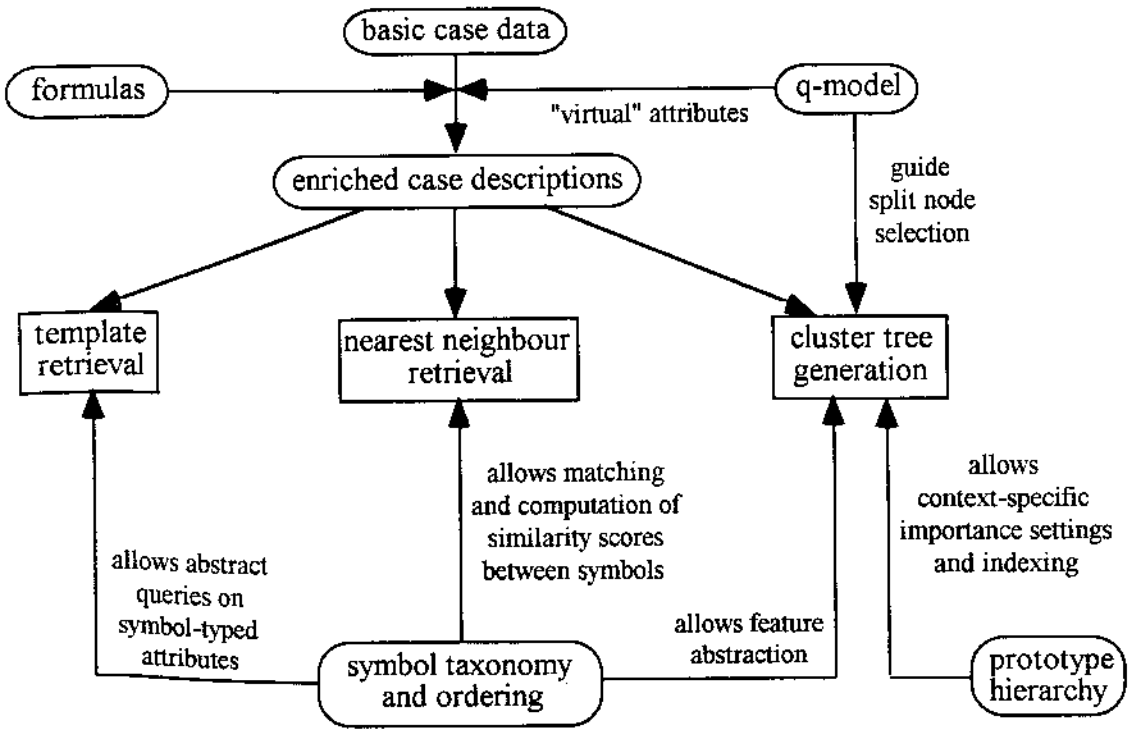

Figure 7.1. Knowledge Representation and Processing in REMIND

In addition, adaptation formulas easily allow retrieved cases to be adapted to fit the data in the new case.

A strong point of REMIND is that it offers two very different similarity assessment methods: nearest neighbour matching and filtering via binary decision trees. Although REMIND does not try to integrate both methods (*i.e.* there is no method for clustering cases according to their "closeness" as determined by a nearest neighbour matching score to improve the efficiency of nearest neighbour retrieval), the different retrieval strategies (including template retrieval) can be "nested" (not automatically but manually). Some combinations may be programmed by using C libraries (Barletta, 1994) to speed up retrieval. However, REMIND cannot be integrated to techniques to cope well with situations where the new case contains missing values that cause the inductive retrieval to come up with a set of cases with different class values, leaving the user the task of finding the most similar case. Instead, REMIND proceeds with a nearest neighbour, or template retrieval, on a set of already-retrieved cases. REMIND is the only shell of those considered that provides the user with the means of retaining an overview of a case base. The database-like queries using templates retrieve all cases that meet a given description and make available basic statistical information (mean and standard deviation for numerical attributes, and number of occurrences of symbols). An automatic testing facility helps validate the accuracy of decision trees.

The same set of cases can be used for more than one problem-solving task. REMIND takes this fact into account by allowing the user to define different local environments, called "views". Case attributes, the cases, the symbol taxonomy plus ordering and formulas computing attribute values are shared by all views; everything else a user defines is view-specific. Each view can have its own q-model, its own cluster-tree (plus prototype hierarchy), its own set of weight vectors, its own adaptation formulas, *etc.* Each view might also consider only a subset of the case base.

REMIND tries to cover as many types of domain as possible. According to the manual, it is intended to provide an environment for developing and using case-based expert systems. The developers of REMIND have tried to include as much functionality as possible into the system as long as neither true programming capabilities, nor a rule interpretation mechanism are required to use them[1]. The only task during the development of a case base that requires programming facilities on the level of defining macros using basic mathematical or text-parsing functions is the definition of formulas. Even so, this is done quite intuitively by "painting" data-flow graphs.

A weak point of the tool is that it seems to have some problems with its memory management - although Cognitive Systems has apparently solved some of these in a later release. In a test using the TRAVEL AGENCY domain, the inductive component was stopped after 36 hours, having corrupted the case memory. This might have been caused by the fact that the system generates binary trees and that there was a large number of different values for the attributes (particularly for the hotel attribute). Better results were achieved by removing attributes with large ranges. Another weak point is the user interface customisation capabilities. We found the form editor to be barely usable. A newer version of REMIND apparently offers links to *Visual Basic* to deal with this specific problem.

---

1 For domains where similarity assessment requires more complex matching methods than nearest neighbour matching, the 'C Library of Functions' would have to be used.

7.5 S³-CASE

The S³-CASE system is a toolbox for solving applications with CBR technology. A graphical editor supports the design of an application in structured domains with many refinable standard types. The graphical user interface gives a good overview of the current application, the case base, the indexing mechanism and the retrieval of new, unsolved problems. The system supports the use of different languages. There is a hypertext interface for describing concepts, attributes, types, *etc.* to archive the used knowledge of the expert during domain design. The environment allows, via a programming interface, the use of the full functionality of SMALLTALK for defining measures. However, to use this powerful feature the designer of a domain must be familiar with the SMALLTALK-80 programming language. The generation of the indexing scheme can be influenced by many parameters for the special needs and wishes of the user. The tool can easily handle flat or structured domains. In particular, S³-CASE had no problems in modelling the structured domain within the evaluation. There is also another interesting test application for supporting the development process of photographs.

The main disadvantage of S³-CASE is the high consumption of memory and disk space. However, for testing we had to use a development version with many irrelevant features in the SMALLTALK image. Most of the time spent in the tests seems to have been spent doing garbage collection. The end user would get a version with lower memory requirements and higher performance.

7.6 Synthesis of the Evaluation

In this sub-section, we highlight some of the results of the evaluation and give an indication of the types of application to which the tools are suited.

In principle, it would have been possible to systematically use all different options of the tools during the tests. However, this would have quickly led to a "combinatorial explosion" of features to test. Instead, we relied on the team member responsible for each tool to decide whether or not the most interesting features of the tools were covered. This demonstrates that such an evaluation heavily relies upon the competence of the evaluation team.

Test T0 (handling structured domains) showed that to handle special kinds of domain it is essential to allow an object-oriented type of representation. From this point of view KATE and S³-CASE appear to have had a certain advantage over the other tools. We feel that the handling of natural domains such as the MARINE SPONGES domain, or more generally, complex and structured domains is a very important aspect.

Tests T5 (compulsory exercise of similarity measures) and T6 (voluntary exercise of similarity measures) made it very obvious that similarity assessment within CBR tools is crucial on the one hand, and on the other, difficult to understand and compare. In test T5, it was possible to be successful using a similarity measure based on incorrect assumptions, such as counting the number of exact matches only. Nevertheless, the test showed that more complicated approaches/systems encountered problems even with simple examples. For this reason, we decided to keep these tests in the evaluation. A further surprise was that none of the tools could take advantage of the exact rule suggested in test T6 as an example.

With reference to the classification of CBR applications given in Chapter 6, Table 7.1 below summarises our opinion about how each tool relates to these applications. A black bullet ● indicates that the tool is likely to be suited to the domain. A white bullet ○ means that the tool's features can be extended in order to cope with the domain. The sign ⊙, given to CBR EXPRESS for help desk applications, reflects the fact that this tool has been specially designed to cope with this particular domain. However, note that Inference now states that help-desk applications ought to be developed using the DDE facilities of Microsoft Windows and integrated into the wider scope of call tracking facilities.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
Classification					
Prediction		○	●	●	○
Assessment	○	●	●	●	●
Help Desk	⊙	○	●	●	●
Diagnosis	●	○	●	○	●
Data Mining			○	○	
In-process Control			○	○	
Off-line Control		●	●	●	○
Synthesis					
Design			○		○
Planning					
Configuration			○		○

Table 7.1 Suitability of tools for different categories of applications from Chapter 5

We can summarise the evaluation as follows: CBR EXPRESS is well-suited for building help-desk systems, REMIND offers many features, KATE 3.0 relies on an induction mechanism that enables it to perform efficient data mining[1], S³-CASE is a powerful tool that is probably best suited to supporting a servicing company and ESTEEM needs further improvements which, according to the supplier, are already forthcoming. None of these tools addresses all the issues. They each provide a good introduction to CBR, but building a CBR application requires additional know-how and skills that go beyond reading a manual. CBR technology is not yet at the level of word processing technology or database management systems. All the successful installed CBR applications of which we are aware required a significant amount of consultancy and adaptation to meet the user's requirements. Developing a CBR application requires investment that goes beyond buying an off-the-shelf product and availability of consultancy is a critical factor before considering any CBR tool. Nevertheless, all the tools we have evaluated provide a good starting point to build a CBR application and can achieve infinitely better results than starting from scratch.

---

1 Supplier's note: KATE 4.0 is much more generic.

Chapter 8: The Evaluation Framework

In this chapter, we present the way we have evaluated the CBR tools and describe the criteria and the application domains. The tests are summarised in Appendix 1 and described in more detail where used in Chapter 9.

The evaluation was structured around two types of criteria - technical and ergonomic. Chapter 9 presents the results of the evaluation based on the technical criteria and Chapter 10 the results based on the ergonomic criteria.

To help assess the tools against the criteria, we used eleven tests based on data from four domains. Obviously, it was not possible to define a test for every criterion.

The tests were all conducted in full at the University of Kaiserslautern and the results were prepared by the evaluation team. One team member was completely responsible for the evaluation of each CBR tool. This included learning how to use the tool, working intensively with the system and the documentation, modelling the four test domains, representing the respective data sets, conducting the tests, selecting relevant criteria for the tool and describing their overall impression. Many tests had to be conducted more than once. We looked at the tools running standalone and did not consider additional programming environments or shells into which the CBR tools could be integrated. We thus excluded systems like Kappa-PC or Art-IM from our evaluation.

8.2 Description of the Evaluation Criteria

There were two main reasons for undertaking an evaluation of software tools. The first reason was to identify problem areas and poor functionality that required improvement: this is of interest to the tool developers and designers. The second was to test for competence and meaningful results in a particular domain: this is of interest to end users. Thus, some basic principles should be defined in order to avoid pitfalls of an inappropriate evaluation. These Principles are as follows:

1. Complex objects or processes cannot be evaluated by a single criterion or number;
2. The larger the number of distinct criteria evaluated or measurements taken, the more available information there is on which to base an overall estimation;
3. Evaluators will disagree about the relative significance of various criteria according to their respective interests;
4. Anything can be measured experimentally as long as exact definitions of how these measurements should be taken are made.

On the basis of these principles, we first developed a framework for expressing a wide range of evaluation criteria to cover most of the theoretical as well as practical aspects of CBR tools (Principles 1 and 2). Secondly, we defined a number of tests to be run on the tools, so that clear-cut measurements could be used to compare the systems (Principle 4). We did not provide overall numerical gradings (Principle 3). Instead, for each criterion we preferred to present a symbolic scale for which the values

were limited to seven possibilities, so that the reader could tell at a glance which system seemed to perform better according to a specific criterion.

We used the following architectural framework to structure our collection of criteria:

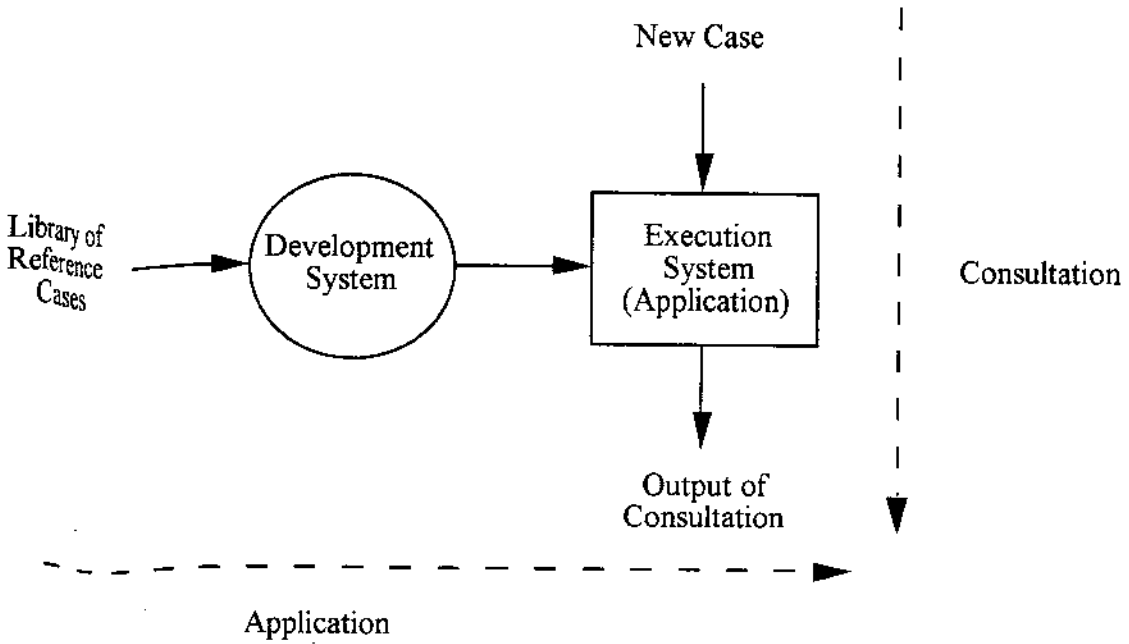

Figure 8.1. The Architectural Framework

From this framework we came up with the following basic terminology:

Development System	This term denotes the functional component for the construction and maintenance of the **execution system**
Application Development	The process of using the development system
Execution System	The application system created using the development system
Consultation	The process of using the execution system

Table 8.1. Important Terms used in this Evaluation

Note that even if the execution and the development systems are one and the same in a particular implementation, we use this distinction at the logical level.

Two major categories of criteria are presented: the technical criteria that highlight the possibilities and limitations of a tool and the ergonomic criteria that deal with how easily these possibilities can be exploited by different kinds of user. Therefore, the technical criteria are mainly related to the execution system, the development system and the case representation, whereas the ergonomic criteria apply to the control of the consultation system and to the application development. For convenience, we structured the criteria as follows (Tables 8.2 and 8.3).

Technical criteria (Chapter 9)		
Case representation (9.1)	*Execution system (9.2)*	*Development system (9.3)*
Describing cases and case bases (9.1.1, 9.1.2) Test T0	Noise and incomplete data in the database (9.2.1, 9.2.2) Tests T1, T2	Noise and incomplete data in the database (9.2.1, 9.2.2) Tests T8, T9
Assessing similarity (9.1.3)	Flexibility (9.2.3)	
Retrieval, reuse, revise and retain (9.1.4)	Performance: speed, memory (9.2.4) Test T3	Performance: speed, memory (9.3.3) Test T10
Representation and use of knowledge (9.1.5)	Correctness, completeness, consistency, effectiveness (9.2.5 - 9.2.8) Tests T4, T5, T6	Consistency, effectiveness, adaptability (9.3.4 - 9.3.6)

Table 8.2. Technical Criteria

Ergonomic criteria (Chapter 10)	
Application development (10.1)	*Consultation system (10.2 - 10.5)*
Control: manual/automatic (10.1.1)	Explainability and modelling support (10.2)
Validation (10.1.2)	User acceptance (10.3)
Data acquisition and maintenance (10.1.3)	Customisation of the user interface (10.3)
	Organisational and human impacts of the technology (10.4)
	Interface to the outside world (10.5)

Table 8.3. Ergonomic Criteria

The above tables are derived from the sections where the results of the evaluations for each criteria are presented, along with which tests (if any) were used. We follow this framework of criteria categories along Chapters 9 and 10, so that for each type of criterion the reader can gain an overview of all systems with respect to a particular aspect, *e.g.* handling noisy data, possibilities of data exchange, *etc.*

To evaluate the systems, we distinguished between three kinds of technique:

1. *Closed* questions (*e.g.* "*What is the quality of the software documentation?*"), that can be answered by a value in the range 0 - 6.

A yes/no question is answered by ● for "yes" and a blank for "no".

In general, the higher the value of a closed question, the more positive the result for the tool. For instance, giving the value 2 to the question "*What is the influence of noisy data on the development system?*" means "noisy data has a rather bad influence on the development system". We indicate the possible answers each time where necessary.

If the result of a test does not agree with the answer to a closed question (*e.g.* a bad result on some tests but a positive answer to the corresponding question), a footnote provides an explanation.

2. *Open* questions (*e.g.* "*What kinds of data types can a system use: symbol, integer, etc.?*").
3. *Tests* that provide clear-cut results in terms of percentages or seconds (*e.g.* the percentage of correct classifications of new cases, or the time needed for case retrieval). However, three tests have qualitative results, namely T0, T5 and T6. This is a result of the context-dependent style of evaluation within these tests.

The tools were evaluated against each criterion through a set of questions (see Table 8.4) and a set of tests. The answers to these questions and tests are based on the four test domains presented below. We therefore have comparison tables similar to Table 8.5.

EXAMPLE CRITERIA TABLE	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ <Question 1>					
<possible answers>	<answer 1>	<answer 2>	<answer 3>	<answer 4>	<answer 5>
scale	4	3	5	*etc.*	
○ <Question 2>					
yes/no		●	●	●	

Table 8.4. Example Criteria Table

EXAMPLE TEST TABLE	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
10%	100%	100%	98%	98%	100%
20%	92%	93%	94%	95%	91%
40%			*etc.*		
60%					

Table 8.5. Example Test Table

8.2 Description of the Test Domains

8.2.1 Car Domain

The CAR domain was created by Jeffrey C. Schlimmer under the original title "1985 Auto Imports Database" in 1987. Most of the data of the CAR domain was taken from Insurance Collision Reports. We chose this domain for three reasons:

- It is freely available;
- It can be understood intuitively;
- It contains a balanced mixture of symbolic and numerical values.

Most of the attributes in the data set characterise cars in their various aspects such as size, type of engine and style. In addition, the set contains an attribute called "symboling" to hold the assigned insurance risk rating. This rating corresponds to the degree to which the car is more risky than its price

indicates. Lastly, every case contains information about normalised losses meaning the relative average loss payment per insured vehicle per year.

Several of the attributes could have served as the target (*i.e.* the class attribute); we decided to take symboling. To be able to interpret the results of the tests correctly, we need to examine the following figure that shows the distribution of values for the slot "symboling":

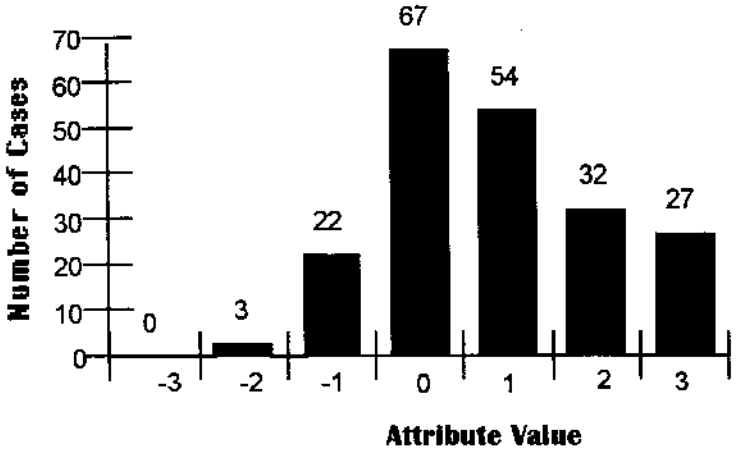

Figure 8.2. Case Distribution for Values of Symboling

The distribution shows that it easy to make a good guess if concentrating on the values 0 and 1. Besides the class attribute, the two attributes "make" and "number-of-cylinders" are not supposed to be discriminant, which means that they should be skipped during similarity assessment.

The CAR domain was used to evaluate the criteria of handling noise and incomplete data in the database, using tests T1, T2, T8 and T9.

8.2.2 CNC Machine Tool Domain

The CNC MACHINE TOOL domain (Althoff, Faupel *et al.*, 1989) is difficult to handle because it contains a large number of unknown values. Furthermore, the overall number of attributes and classes is very high.

The main reason we chose this domain was the experience we gained on it during the development of *Patdex*, an integrated CBR system developed at the University of Kaiserslautern (cf. Althoff & Wess, 1991).

In fact, thanks to our extensive comprehension of the CNC MACHINE TOOL domain, we were able to dynamically change sub-parts of the respective test. This allowed us to focus on the features of the tools being evaluated, with respect to these tests, resulting in more helpful statements on similarity assessment for these tools.

The CNC MACHINE TOOL domain was used to evaluate the criteria of assessing similarity, using tests T5 and T6.

8.2.3 Marine Sponges Domain

This database was derived from an application developed at the French National History Museum where the aim was to help biologists classify marine sponges. There are nine thousand different species of sponges described in three classes (*Calcarea*, *Hexactinellida*, *Demospongiae*).

The main issue with this domain was how to represent the expert knowledge. It appeared to be necessary to build a database that represented the natural variety and richness of example sponges. This would mean all expert knowledge had to be expressed via the description language (*e.g.* relationships between sub-parts of sponges, uncertainty on measures, taxonomy on some attributes, *etc.*).

The MARINE SPONGES domain was used to evaluate the criteria concerned with describing cases, using test T0.

8.2.4 Travel Agency Domain

This domain was first introduced by Mario Lenz who developed the *CABATA* (Case Based Travel Agency) system. The data used in the system was taken from ordinary travel brochures. Mario Lenz added domain knowledge in different representations to make the *CABATA* system run successfully. We removed this additional knowledge to make sure that each tool could handle the domain. This did not constitute a restriction since we used the domain purely for performance testing. The domain is well-suited for such tests in that the number of cases is fairly high and the success of a retrieval step can be judged without any deep expert knowledge.

Each case in the TRAVEL AGENCY domain is made up of nine attributes: comfort, duration, holiday type, number of persons, region, price, month, transportation and hotel name. We added a case-number to make it possible to compare different retrieval results. Neither this additional attribute nor the hotel name were used for retrieval.

The TRAVEL AGENCY domain was used to evaluate the criteria concerned with performance - speed of case retrieval and index construction - using tests T3 and T10. It was also used for the correctness and consistency criteria using tests T4 and T7.

8.2.5 Summary of Test Domains

	MARINE SPONGES	TRAVEL AGENCY	CNC MACHINE TOOL	CAR
# of cases	121	1,470	311	205
# of classes	low	none	high	low
# of attributes	35	9	80	26
# of numeric attributes	10	3	0	15
# of values per attribute	few	many	few	few
% unknown values	low	none	high	low
Can the data be represented as flat tables?	no	yes	yes	yes
Background knowledge	yes	yes	no	no
Specific features	complex case structures	many cases	many unknown attribute values	public domain

Table 8.6. Overview of the Four Test Domains

Chapter 9: Evaluation based on Technical Criteria

In the first section of this chapter we cover the criteria that deal with the underlying case and knowledge representation of each of the CBR tools. In the second and third sections we cover the criteria that relate to the execution system and the development system respectively.

Under some of the criteria, some of the evaluators have chosen to include detailed comments relating to the tool(s) they evaluated.

9.1 Case and Knowledge Representation

In this section, we cover all representational aspects of the CBR tools. We start with the representation of cases and their organisation within the case base, then go on to the assessment of similarity between cases, followed by the reuse, revision and retention of cases. Finally, we cover the representation and use of background knowledge.

9.1.1 Describing Cases

In this sub-section we concentrate on the expressiveness of the case representation language. We distinguish between available and definable attribute types, the former referring to predefined data types offered by the tool, the latter referring to user-definable, user-refinable, or composed types. Whereas an attribute of type *Case* contains a reference to another case in a case base, an attribute of type *Object Hierarchy* supports modelling of structured domains that require inheritance, such as the MARINE SPONGES domain.

9.1.1.1 Representation of Structured Domains

TEST T0:

Ability to handle structured domains

As a first test, we determined to what extent the CBR tools were able to model the MARINE SPONGES domain. Due to the complexity of the domain, this task is not trivial. Firstly, sponges are not represented as flat structures but as nested objects. Secondly, this domain needs attributes that can store one or more values. In particular, most of these multi-valued attributes contain complex objects, so that they cannot be represented as lists of primitive types.

To deal with these problems, the conclusion is that a fully structured, object-oriented language is necessary, allowing hierarchical descriptions involving complex sub-objects and relationships between these sub-objects. The only systems that support such a description language are KATE and S^3-CASE,

since they support the CASUEL object-oriented language. The other systems do not support object hierarchies and relations (cf. also Table 9.2).

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ Can the marine sponges domain be represented? *yes/no*			●		●
○ Is it possible to use complex data represented as objects? *scale*		2[1]	6	1[2]	6

Table 9.1. Results of Test T0

[1] ESTEEM is able to represent object-oriented structures as 'nested cases'. Each slot of a case can contain another case which may have a different structure. This nesting mechanism is limited to 5 levels.

[2] Values of type *Case* might be cases from the currently considered case base, or from another, already existing one. However, matching on attributes of this type during nearest neighbour retrieval will be a simple identity check of the case-ID's, instead of using some similarity measure defined in the case base to which the referenced cases. The only way to include attributes of the referenced cases into the similarity scoring is to access them via formulas and to store the values into new attributes added to the referencing cases. This only works with non-symbol attributes.

9.1.1.2 General Expressiveness of the Tools

Table 9.2 shows the general expressiveness of the tools. Pre-defined or definable attribute types are listed, together with the multi-media capabilities.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ Which attribute types are available?					
Boolean	●	●	●	●	●
Integer			●	●	●
Real	●	●	●	●	●
Date				●	
Time				●	
String	●	●	●	●	●
Symbol	●	●	●	●	●
Ordered Symbol			●	●	●
Symbol Taxonomy			●	●	●
Case		●		●	
List			●	●	●
Relations			●		●
○ Which attribute types are definable?					
Sub-range	●	●	●	●[1]	●[2]
Conjunction					●
Disjunction					●
Enumeration			●	●[3]	●
Interval		4	●		●
Multi-Value		5	●	●	●
Object Hierarchy			●		●
○ Which media can be used for additional information?					
graphics	●	6	●		●
sound	●	6	●		●
video	●	6	●		●
○ Is it possible to use default values?					
yes/no			●	●	●

Table 9.2. Case Representation I

1 Subranges are supported for Date, Time, Real and Integer.
2 It is possible to restrict the allowed values by an enumeration or a range specification for all types.
3 Enumeration of symbols only.
4 Supplier's note: ESTEEM 1.4 supports enumeration and interval by allowing a fixed range for numeric entry. When the user chooses <numeric> as a feature type, he is automatically asked for numeric constraints.
5 Supplier's note: ESTEEM 1.4 supports multivalued features. Two kinds of similarity assessment are possible: the superset feature matching (sim((a, b, c), (a, b, d, e)) = 2 / 5 = 40%) and the subset feature matching (sim((a, b, c), (a, b, d, e)) = 2 / 3 = 66%).

9.1.1.3 Other Specific Features

Table 9.3 covers some specific features that appear to be important. In particular, the handling of unknown values is critical for CBR tools, since generally only a partial description of the new case is given to the system.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ Is it possible to handle missing values (*do not care*)?					
explicit handling		●	●	●[1]	●[2]
implicit handling	●[3]				
○ Is it possible to handle unknown values (*not known*)?					
explicit handling	[4]	●	●	●	●
implicit handling					
○ Is it possible to use uncertain values?					
yes/no		[5]	●	●	●
○ Is it possible to use negative examples?					
yes/no					

Table 9.3. Case Representation II

Note that the use of uncertain values can be handled in two different ways: indirectly, it is possible to define taxonomies to represent a partial unknown for an attribute. For example, if you do not know the exact colour of an object, you can define *dark colour* as a value for different dark colours like *black, grey, brown etc.* KATE, REMIND and S³-CASE allow this possibility. The second possibility is to define the colour as a multivalued attribute with a restricted cardinality. This facility is available in KATE and S³-CASE.

---

6 Supplier's note: ESTEEM 1.4 provides graphics, sound and video through the definition of a <multimedia> feature type. The user then provides a path name for a multimedia file which is used as part of the case.

1 The user can choose one of three different global methods of handling missing values: 1) perfect match, 2) complete mismatch, 3) ignore the attribute.

2 It is possible to give a zero weight for do-not-care attributes for special cases with rules, such that they no longer influence the computation of similarity.

3 Cases may consist of different numbers of questions.

4 Unknown is a possible slot-value, but a case with unknown values cannot be used for retrieval.

5 Supplier's note: ESTEEM 1.4 offers the possibility to treat uncertain values with the supertype <inferred>.

9.1.2 Organisation of the Case Base

ESTEEM: An index can be built with some attributes that have to be selected manually. The indexing method used by ESTEEM is not documented in the manual. Our experience with the tool has led us to the conclusion that the indexing structure is probably a hash table with a key derived from the selected attributes. This method collects all cases with the same value for these attributes in categories strictly separated from each other. The nearest neighbour retrieval is constrained to one of these categories, defined by the values of the new case, omitting the rest of the case base which still may contain similar cases.

REMIND: The automatic indexing scheme used by REMIND is supported by background knowledge from experts. The following is a short description of the process:

REMIND can create binary decision trees, using a clustering algorithm from statistics (CART algorithm, see section 3.1.3) for use as an automatic index. The manual refers to Breiman, Friedman *et al.* (1984). Only attributes specified by the user as relevant will be taken into account during this indexing process.

There are two means of influencing the automatic indexing scheme used for building the cluster tree (not counting the bucket-size parameter that determines the minimum number of cases in a cluster that makes a further split necessary): defining prototypes and defining a q-model (see section 7.4).

- Prototypes are user-defined AND/OR formulas of attribute relations, intended to reflect a description of a prototypical case. An attribute relation is a binary relation between a case attribute and a constant value of that attribute's value type (*e.g.* "age < 18" if "age" is the name of an integer-typed case attribute). For each type, there is a fixed set of possible relations.

A prototype hierarchy can be used to pre-filter the case base, since all cases will be indexed under those prototypes that describe them. The cases indexed under such a prototype can then be further indexed using the automatic indexing scheme, with the possibility of using different parameters for each prototype.

- The q-model is intended to reflect causal relationships between case attributes. This knowledge about causal relationships can be used during the construction of the cluster tree, in that it defines a preference ordering on the set of attributes, restricting the choice of the attribute on which to base the next split (*i.e.* the construction of a new decision node), since, in general, an attribute that influences the value of other attributes should be considered as a split candidate first.

REMIND offers different methods for retrieval, including a basic mechanism to allow simple database-like queries, called template retrieval. Templates are user-definable AND/OR formulas of attribute relations, forming a search pattern that the retrieved cases have to meet. Templates can be seen as *temporary* prototypes; retrieval will return all cases matching the template specified. Template retrieval is of invaluable help for maintaining or exploring case libraries.

REMIND also allows the three different strategies to be "nested" in a certain way. For example, one can first retrieve a set of cases via inductive retrieval, proceed with a nearest neighbour search on this set and finish with a template retrieval on that search's result. This can be useful in situations when, due to missing values in the new case, inductive retrieval returns cases with different class values. However, this nesting always has to be carried out manually in that it is not possible to program a fixed succession of retrieval steps.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ Which indexing schemes are available?					
manual indexing			●	●	
automatic indexing	●	●	●	●	●
decision tree			●[1]		
k-d tree					●
cluster tree				●	
○ Which indexing schemes are used by default?					
automatic indexing	●				
linear list		●		●	
○ It is possible to further influence the indexing?					
yes/no			●	●	●
○ Which retrieval techniques are available?					
nearest neighbour	●	●	2	●	●
template retrieval				●	
static inductive			●	●	
dynamic inductive			●		

Table 9.4. Organisation of the Case Base

9.1.3 Assessing Similarity

There are two different approaches to assessing similarity: exact matching against an abstract description (*e.g.* automatically extracted knowledge stored as a decision tree), or nearest neighbour matching using a numerical similarity measure. The KATE tool employs the first approach to similarity assessment. CBR EXPRESS, ESTEEM and S³-CASE allow the adjustment of the similarity assessment methods, but to different degrees. Users of REMIND can choose between both approaches. The tools offering nearest neighbour matching allow the assignment of importance for each attribute in the form of weights. Some tools are able to determine a similarity score taking into account only sub-parts of complex cases, called local contexts. Filtering means reducing the whole case base to a conclusion by traversing a decision tree until a conclusion is reached. This can be regarded as exact matching on an

1 The decision tree in KATE-INDUCTION is static whilst KATE-CBR calculates the tree dynamically.
2 Supplier's note: nearest neighbour matching is available in KATE 4.0.

abstract level. Static filtering using a fixed decision tree contrasts with dynamic filtering, where reduction takes place incrementally. Preferences allow the explicit ranking of cases using rules.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ Which similarity assessment techniques are available?					
nearest neighbour	●	●	1	●	●
local context		●		●	●
assig. of importance	●	●	1	●	●
preferences			1		●
static filtering		●	●	●	●
dynamic filtering			●		
programmable					●
○ Can the similarity measure be modified automatically?					
yes/no		2			●
○ How does the similarity measure handle numeric values?					
exact match		●		●	●
linear distance	●[3]	●[4]	●		●
programmable		5			●
○ How is the similarity measure determined between symbol values?					
exact match	●	●	●	●	●
user-defined			1	●	●
programmable		5			●
○ How is the similarity measure determined between strings?					
exact match			●	●	
built-in matching	●[6]	●[7]			
programmable		5			●

Table 9.5. Assessing Similarity I

1 Supplier's note: KATE 3.0, used for this evaluation, did not include these options. KATE 4.0 does.

2 Supplier's note: ESTEEM 1.4 offers the possibility to adapt the weights used in the similarity assessment (see Table 9.6).

3 Within a 10%-Interval.

4 Within a definable Interval.

5 Supplier's note: ESTEEM 1.4 provides many more ways than listed in the table.

6 Special string matching (skipping of fill-words, stable against misspelling or changed ordering).

7 Hamming distance can be switched between characters and words, case sensitive or non-sensitive.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ What kinds of weights are used?					
case-specific	●				
attribute-specific	●	●[1]	●	●	●
relevance					●
match/mismatch	●				●
unknown values				●[2]	●
global parameters	●				
programmable		3			●
○ Is it possible to use knock-out attributes?					
explicit	●		●	●	●
implicit		●			
○ Is it possible to use thresholds?					
one		●[4]			
two	●				
more					●
○ Is it possible to incorporate declarative knowledge?					
scale		3		5	4

Table 9.6. Assessing Similarity II

9.1.4 Reusing, Revising and Retaining Cases

As can be seen in Figure 2.1 (Chapter 2), reusing, revising and retaining cases are valuable features in the CBR cycle. They are necessary in order to make the tool incremental. Note that incremental operation may be seen as an advantage - for example, taking new entries directly into account in a process control operation - or as a disadvantage - for example, when a help-desk system has to be validated before it is delivered onto a network.

Table 9.7 shows that only a few tools employ techniques for handling these valuable properties. ESTEEM uses rules, REMIND proposes adaptation formulas and S³-CASE an update of its relevance matrix of weights in case of failure.

---

1 Can be dynamically computed by using of rules.

2 Three different matching methods are available: i) perfect match, ii) complete mismatch, iii) ignore the attribute. For matching against prototypes, the handling of missing values can be attribute-specific.

3 Supplier's note: ESTEEM 1.4 allows the system to automatically determine weights using ID_3 or gradient descent algorithms (see Section 3.1.1), or the user can determine them using rules.

4 ESTEEM always uses a definable (default 50%) threshold to determine, whether two cases are similar enough.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
❍ Does the tool employ techniques to improve the reusing of cases?					
yes/no		●		●	
❍ Does the tool employ techniques to handle case revision?					
yes/no					●
❍ Does the tool employ techniques to handle case retention (storage of new cases)?					
yes/no	●	●	●	●	●

Table 9.7. Reusing, Revising and Retaining Cases

9.1.5 Extracting, Representing and using Background Knowledge

Additional background knowledge is helpful in many situations. For example, it can be used to restrict the search space when looking for the solution of a diagnosis problem. In Table 9.8 we give an overview of various types of background knowledge used in the CBR tools.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
❍ Which extracting techniques are available?					
categories				●[1]	
weights					●
decision trees			●		
heuristic rules			●		
❍ What kind of rules can be used?					
causal		●			●
❍ What kind of further knowledge can be used?					
sub-range definitions	●	●	●	●	●
symbol taxonomies			●	●	●
symbol orderings				●	●
object hierarchies			●		●
decision trees			●	●	
formulas				●	
qualitative models				●	
prototype hierarchies				●[2]	
❍ Where is background knowledge used to improve the CBR process?					
retrieval		●	●	●	●
reuse		●		●	

Table 9.8. Extracting, Representing and Using Background Knowledge

1 In the form of clusters indexed by a cluster tree.
2 See Section 7.4 for a description of q-models and prototypes.

9.2 Execution System

In this section we cover the performance of the CBR tools during the consultation phase. We investigate the ability to handle noisy and incomplete data and consider issues such as flexibility, performance, correctness, completeness, consistency and effectiveness.

9.2.1 Ability to handle Noisy Data during Consultation

TEST T1

Systematic introduction of noise during consultation

For this test we used the CAR domain to introduce noise incrementally into the new cases. All 205 cases of the domain were taken as reference cases and used for building an execution system. From the 205 cases, 50 were randomly selected as new cases. Then we introduced noise by randomly selecting a possible value from the respective value range for 10%, 20%, 40% and 60% of the attributes, respectively. According to the different results of the systems, we computed the percentage of correct classifications for each degree of noise. In the following paragraphs we give a short explanation of how the test was conducted with each tool.

CBR EXPRESS: The weights for **CBR EXPRESS** were initially set to 1 for non-discriminant attributes, which is the minimum from a scale up to 100 (see also a comment on this point at the end of section 7.1). All discriminant attributes kept the default weight of 10. To get reasonable results, we retrieved the best 5 cases on each new case. The answer was judged as correct only if all retrieved cases within a bandwidth of 2% came up with the right diagnosis. Finally, it should be mentioned that, even when noise was increased up to 60%, the tool managed to find the *identical* case in 38% of all queries.

ESTEEM: In this test, as well as in tests T2, T8 and T9, **ESTEEM** used a similarity measure that weighs all attributes equally. Only exact matches were counted, although it is also possible to use linear distance measures for numeric attributes.

As **ESTEEM** is not able to restrict the number of retrieved cases, we used the first one in the list of retrieved cases as the result, even if there were more retrieved cases with the same similarity score. In such a situation, the first case shown may contain an incorrect classification, whilst the second or third one, still with the same similarity score, might be the correct result. Therefore, even if the correct result was amongst the cases with the highest score, we counted this as a bad classification, because we only looked at the very first retrieved case. Note that the same problem occurred in REMIND.

KATE: The test was conducted with the KATE-INDUCTION component. After consulting the decision tree, the user gets a probability distribution for the different diagnoses. If the probability for the right diagnosis was higher than for all other diagnoses, this was taken as the classification result. Due to the nature of decision trees with their sensitivity to noise, the results are not so good as compared with the tools using nearest neighbour matching; a single changed attribute value can prevent the original case from being found.

REMIND: This test, as well as tests T2, T8 and T9, was conducted using nearest neighbour retrieval with identical weighting of all attributes; missing values were ignored for the computation of the similarity score. The classification of "the" nearest neighbour was compared with the correct classification. It should be mentioned that a random element enters into the test results if two or more cases have the same maximal similarity score. The test data for REMIND do not take account of this; as with ESTEEM, the system was asked to retrieve one nearest neighbour. We consider it to be a weak point if a tool does not warn/inform the user about the existence of other cases with the same similarity, that are simply not shown because the number of cases to be retrieved had been arbitrarily restricted by the user.

In this test (as well as in test T8), REMIND becomes the victim of its carefully computed similarity score on numerical values. In fact, the surprising result of this test is not the fast increase of bad classifications by REMIND, but rather the correct classifications by the other tools applying nearest neighbour matching. An example will illustrate the point:

attributes	case152 (noised)	case152 (original)	case154
engine location	rear	front	front
height	54.5	54.5	59.1
length	158.7	158.7	176.58
symboling (target)	one	one	zero

Table 9.9. Problems with Test T1

The table contains only part of the case descriptions used in the CAR domain, but let us assume those three attributes would make up the whole case description (the target slot "symboling" is, of course, not taken into account). The similarity score between case152 (noised) and case152 (original) will always amount to 66% because of the difference in the symbol-typed attribute "engine location". Yet, the similarity score between case152 (noised) and case154 will depend on the actual distribution of the values for "height" and "length" in the case base, due to the normalisation method employed by REMIND; it could be lower or higher than 66%. This is what seems to have occurred when REMIND performed its bad classifications when doing nearest neighbour matching with a noised case. Whilst the total number of attributes with different values was usually higher in the retrieved cases than in the (non-best-scoring) original cases, the number of symbol-typed attributes that had different values was usually higher in the original case than in the retrieved cases. Since no ordering or taxonomic information

on symbols had been defined, different symbol values always resulted in a complete mismatch on such an attribute. The performance of the other tools in this test could simply be explained by them having "punished" differences in numeric values most of the time as hard as they "punished" differences in symbolic values. Using exact matches, the results could have been improved (see ESTEEM).

S³-CASE: The test was conducted with the standard indexing parameters of the *k*-d tree. After consulting the S³-CASE system with the new cases, the user gets a list of the retrieved cases ordered by their similarity. Only if all cases with the maximum similarity contained the same right diagnoses was this considered to be a correct classification.

	CBR EXPRESS	ESTEEM	KATE 3.0 induction	REMIND nearest neighbour	S³-CASE
10% noise	100%	100%	84%	98%	100%
20% noise	98%	100%	70%	90%	98%
40% noise	92%	94%	54%	66%	92%
60% noise	88%	72%	40%	42%	82%

Table 9.10. Test T1

9.2.2 Ability to handle Incomplete Data during Consultation

TEST T2:

Systematic reduction of available information within the new case during consultation

All the tools used the same test procedure as in test T1.

One advantage of CBR is the ability to make decisions using incomplete information. The goal of this test was to find out how much information was needed by the different tools to get a correct classification result.

The reference case base for this test consisted of all the 205 cases of the CAR domain. From this set, 50 cases were selected randomly to represent the new cases. The information in the new cases was then reduced by deleting a given percentage of the attribute values in each case. We made this deletion four times using 10%, 20%, 40% and 60% as a parameter. The attributes chosen were independent for each run.

The resulting new cases were then entered into the CBR tools to retrieve the most similar case for each one. The "symboling" attribute was used as the classification result and, thus, did not take part in the similarity assessment.

The result of this test is the percentage of correctly classified new cases for each degree of information reduction.

As the new cases were built from cases that were present in the reference case base, there was always a case with a maximum similarity score and the correct classification. We might therefore expect a classification result of 100% for all tools. However, it was possible that the remaining information in a new case matched several reference cases that could have a different "symboling" value. As the tools were advised to retrieve only one case, they had to choose one of possibly several cases with a maximum similarity score. In that circumstance, a CBR tool produce an incorrect classification, which partly explains test results that do not reach 100%.

ESTEEM: The similarity measure used in this test calculated the percentage of exactly matching attributes, based on the total number of possible attributes and not just on the number of given attributes. Thus, the highest possible score for a new case with 60% reduced information was 40%. As the resulting percentages were rounded to integer values, ESTEEM could compute the same similarity score for cases with a slightly different similarity to the new case. In particular, there were more cases with a maximum similarity score than without rounding. As only one of these cases could be presented as the most similar one, the probability for a bad classification was higher than without the rounding, which can be seen in the test results for 40% and 60% information reduction.

KATE: The reduction of correct classifications was caused by the increase in the number of cases found during decision tree traversal. As more cases were retrieved, the outcome of the consultation increasingly reflected the distribution of the diagnoses in the whole case base.

REMIND: Since missing values were ignored, the original of the new case was always amongst the best-scoring cases. The two bad classifications that occurred with 60% information reduction are due to the random effect mentioned in the comment about test T1.

	CBR EXPRESS	ESTEEM	KATE 3.0 induction	REMIND nearest neighbour	S³-CASE
10% reduced	100%	100%	92%	100%	100%
20% reduced	100%	100%	80%	100%	100%
40% reduced	100%	78%	72%	100%	100%
60% reduced	100%	82%	54%	98%	98%

Table 9.11. Handling Incomplete Data during Consultation (Test T2)

9.2.3 Flexibility of the Execution System

In the following, different kinds of consultation mode appear: system-driven, system/user-driven, user/system-driven, user-driven. A system-driven consultation completely controls the order of interactions with the user by prompting for answers, in contrast to a user-driven consultation, where the order of attribute value specifications is completely determined by the user. An example of the former is

dynamic case-filtering; an example of the latter is nearest neighbour retrieval. The other two kinds represent combinations of both consultation modes, where the mode order reflects the degree of control.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ Is the user able to correct previous input during interactive consultation?					
scale	4[1]	6	6	6	6
○ Is the system able to use additional strategies of problem solving?					
dynamic case-filtering			●		
clustering				●	
○ Is the system able to confirm or deny a solution when the user asks for that one?					
yes/no				●[2]	
○ Does the system handle a refutation of the solution by the user?					
yes/no					●[3]
○ Does the system support different modes of consultation?					
system-driven			●		●
system/user-driven			●	●	
user/system-driven	●		4		
user-driven		●	4	●	●
○ User-driven consultation: how many attributes can be selected at once?					
all		●	4	●	●
some	●				
one			●		
○ System-driven consultation: does the execution system control the flow of the consultation by prompting the user for input?					
yes/no			●	●	●

Table 9.12. Flexibility of the Execution System

1 The possibility of correcting previous input is restricted. The questions supplied to the user are collected from the list of currently matching cases whilst the most promising question is presented first.

2 At the end of an inductive retrieval, REMIND not only returns the set of cases found during cluster tree traversal, but also comes up with an 'explanation' of its result. This explanation consists of an overview of which prototypes applied, which decisions (i.e. tests on attribute values) were made during tree traversal until it encountered a case cluster, which different outcome values have so far been found and the number of their occurrences etc. There is no possibility of denying a solution.

3 The system allows cases with certain classification values to be excluded from those retrieved. Refutation can be achieved by preventing the same classification being proposed again (in subsequent retrievals during the current consultation).

4 Supplier's note: these options are available with KATE 4.0.

9.2.4 Performance of the Execution System

KATE: There are different results in the speed of consultation between the induction component and the CBR component of the system. Using induction, the decision tree is generated during the building of the execution system. Thus, there are short consultation times. In the CBR mode, the case base is used to dynamically calculate the best attribute for filtering the case base. For a large number of attributes and a large number of cases in the case base, this can take a few seconds for each attribute whose value is requested.

REMIND: The test was conducted using nearest neighbour retrieval only, since it would not make any sense to cluster the case base and carry out inductive retrieval without any useful target attribute. The rather long time required by REMIND is probably caused by its complex symbol matching method during the linear search (notice the high number of symbol-typed attributes in the TRAVEL AGENCY domain). Since we had to remove the symbol taxonomy originally connected with the domain (*e.g.* to cope with general descriptions of the kind of place a client wants to travel to) so that all tools could handle the domain, REMIND does this work unnecessarily.

S^3-CASE: The high values for consultation with 1,420 cases resulted from the large memory requirements of the S^3-CASE system. The real consultation time is almost linear with the number of cases in the case base, but during the consultation, much of the time was taken up with SMALLTALK'S garbage collection. Thus, the time for garbage collection has this negative effect on average consultation time.

TEST T3:

Retrieval Time

The same settings as in tests T1 and T2 were used.

The TRAVEL AGENCY domain was chosen for this test because it encompasses the highest number of cases. 50 of the 1,470 cases were used as new cases. The remaining 1,420 cases were partitioned into four sets: two sets of 200 cases, one of 400 and one of 620 cases. By successively merging these partitions, we obtained reference case bases of 200, 400, 800 and 1,420 cases.

The times were measured for retrieving the most similar case and the five and ten most similar cases for a new case. These values were measured on the same hardware with no other program running on it, except the CBR tool itself and the Windows Program Manager. The values shown in the table below are the average values over the 50 new cases for every combination of the parameters. Since the time was measured manually, the values may include a few tenths of seconds of "reaction time", which can slightly falsify the results. Our definition of "retrieval time" used in this context is the time elapsed between triggering the retrieval with a mouse or keyboard action and the displaying of the results on the screen.

These tests were carried out without building an index, because building an index is a special feature in ESTEEM and not a default setting. Therefore, it is not surprising that the retrieval time grows almost linearly with the size of the reference case base, as the tool probably does a linear search.

	CBR EXPRESS	ESTEEM[1,2]	KATE 3.0	REMIND	S³-CASE
most similar case of 200 cases	1.0 sec.	7.1 sec.	< 1 sec.	5.2 sec.	1.5 sec.
five most similar cases of 200 cases	1.1 sec.	7.1 sec.	< 1 sec.	6.8 sec.	1.5 sec.
ten most similar cases of 200 cases	1.1 sec.	7.1 sec.	< 1 sec.	7.4 sec.	1.5 sec.
most similar case of 400 cases	1.1 sec.	14.0 sec.	< 1 sec.	8.4 sec.	3.1 sec.
five most similar cases of 400 cases	1.1 sec.	14.0 sec.	< 1 sec.	11.7 sec.	3.1 sec.
ten most similar cases of 400 cases	1.1 sec.	14.0 sec.	< 1 sec.	12.5 sec.	3.1 sec.
most similar case of 800 cases	1.2 sec.	26.1 sec.	< 1 sec.	13.6 sec.	6.8 sec.
five most similar cases of 800 cases	1.2 sec.	26.1 sec.	< 1 sec.	18.0 sec.	6.8 sec.
ten most similar cases of 800 cases	1.3 sec	26.1 sec.	< 1 sec.	19.6 sec.	6.8 sec.
most similar case of 1,420 cases	1.3 sec	46.9 sec.	< 1 sec	29.3 sec.	64 sec.
five most similar cases of 1,420 cases	1.4 sec.	46.9 sec.	< 1 sec.	38.1 sec.	64 sec.
ten most similar cases of 1,420 cases	1.4 sec.	46.9 sec.	< 1 sec.[3]	42.6 sec.	64 sec.

Table 9.13. Speed of the Execution System (Test T3)

1 In ESTEEM, the user cannot limit the number of retrieved cases to a special value. Therefore, the table shows three times the same value for each size of the reference case base. With the same settings, the pure retrieval time does not change significantly from one new case to another. However, the time needed to build the result window depends strongly on the number of cases actually retrieved. An empty result window appears almost at once, whereas a window containing 10 cases may take about 4 seconds to be displayed after the "please wait" message has disappeared.

2 Supplier's note: ESTEEM 1.4 employs newly coded retrieval algorithms that would change these times significantly.

3 Supplier's note : although the results presented for KATE are better than with the other systems, AcknoSoft would like to state that it feels the test is not relevant and comparable to the other systems. Indeed, the retrieval time was tested for retrieving a case in a decision tree because KATE 3.0 did not include nearest neighbour. KATE 4.0 includes a nearest neighbour retrieval with pure unindexed nearest neighbour. Retrieval time between 1 and 2 seconds has been achieved for the 1,470 cases with KATE 4.0.

Table 9.14 shows the requirements of main memory and disk space. The given values were either taken from the documentation, or they were verified during the tests (TRAVEL AGENCY domain / 1,470 cases).

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ What is the consumption of main memory of the execution system necessary to handle the respective domains?					
	8 (Min. 4) MB	8 (Min. 4) MB	4 MB	8 MB[1]	16 MB
○ What is the consumption of disk space of the execution system necessary to handle the respective domains?					
	900 KB	700 KB (with index: 1.4MB)	10 KB[2]	1 MB[3]	13.6 MB

Table 9.14. Main Memory and Disk Space Requirements

9.2.5 Correctness of the Execution System

TEST T4

Classification of Identical Cases

When the new case is an exact copy of an existing case in the case base, we would expect a CBR system to recognise this and to retrieve the corresponding case. In order to test whether the CBR tools do this, we used the same test configuration as in test T3, but this time used the whole base of 1,470 cases as reference cases. This ensured that every new case was also present in the case base. Again, we retrieved the most similar case for each of the 50 new cases. As there were no duplicates in the case base, the most similar case had to have the same ID as the new case. The result shown in the table below was the percentage of correctly retrieved cases.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ Does the execution system behave correctly with respect to the input data and the underlying similarity measure?					
scale	5[4]	6	6	6	6
○ TEST T4					
	100%	100%	100%	100%	100%

Table 9.15. Correctness of the Similarity Measure (Test T4)

1 Although after loading the case base REMIND only required about 3.5 MB, this value increased whilst working with the case base..

2 The value given in the table only refers to the decision tree when explicitly stored on disk.

3 The value given in the table only refers to the case base; the REMIND development shell needs about 2.8 MB on a hard disk. We do not know how much a runtime version of REMIND would require.

TEST T5

Compulsory Exercise of Similarity Measures

Here we tested the capabilities of the similarity measures using the default settings. We used the following cases that were structured according to the CNC MACHINE TOOL domain, but which were not present in the existing case base. In the first step of this test, CaseCorrect and CaseIncorrect made up the complete reference case base; for the second step, the 311 original cases plus these two cases made up the reference case base. The new case was the same in both steps.

CaseCorrect:
ErrorCode = i59
I/OStateOut7 = logical-1
Valve5Y1 = switched
I/OStateOut24 = logical-0
Valve5Y2 = not-switched
PipesClampingReleaseDevice = OK
I/OStateIn32 = logical-1
DIAGNOSIS = IOCardIN32i59Defect

CaseIncorrect:
ErrorCode = i59
Valve5Y1 = not-switched
I/OStateIn32 = logical-1
DIAGNOSIS = MagneticSwitch5Y1Defect

New case:
ErrorCode = i59
Valve5Y2 = not-switched
I/OStateIn32 = logical-1

We assume that the correct diagnosis for the new case is *IOCardIN32i59Defect*.

The new case matches CaseCorrect in all of the three given attributes, whereas it has only two common values with CaseIncorrect. On the other hand, it does not contain values for four attributes filled in CaseCorrect and only one for CaseIncorrect. In this situation, a system like PATDEX (see section 8.2.2) will always retrieve CaseIncorrect, *i.e.* the incorrect result, as the most similar case, because it assigns a negative weight to missing values.

Table 9.16 shows the result for the tools examined in this test.

4 Changing weights for numerical values can cause problems - see Section 7.1 for explanation.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
most similar case	CaseIncorrect	CaseCorrect	CaseCorrect	CaseCorrect and CaseIncorrect[1]	CaseCorrect
cases with the two highest degrees of similarity	CaseIncorrect (65%), CaseCorrect (43%)	CaseCorrect (4%), CaseIncorrect (3%)	CaseCorrect (100%), CaseIncorrect (0%)	CaseCorrect (100%), CaseIncorrect (100%)[2]	CaseCorrect (2.6%) CaseIncorrect (2%)

Table 9.16. Results of Test T5

TEST T6:

Voluntary Exercise of Similarity Measures

This test is a kind of free-style exercise to examine various features of the CBR tools. We tried different similarity measures and discuss below their effect when applied to the same data. The common data for this test consisted of the whole CNC MACHINE TOOL case base with its 311 cases plus the cases from test T5. The new case was also the one from test T5.

The value of ErrorCode in CaseCorrect and CaseIncorrect, i59, does not appear in the rest of the case base. However, there is another case, Case138, with the correct diagnosis *IOCardIN32i59Defect*.

As a common start, all the tools had to repeat the query from test T5 with the larger case base, but with the same similarity measure. The question was: were other cases, in particular Case138, retrieved? Then we tried to improve the result by adding different features to the similarity measure. We hoped we could include the use of background knowledge, for instance as expressed in the following causal rule:

IF ErrorCode = i59 AND Valve5Y2 = not-switched

THEN I/OStateOut24 = logical-1

Unfortunately, it appeared that none of the tools could make explicit use of this rule (although it was possible to represent it in REMIND and S³-CASE) to retrieve CaseCorrect instead of CaseIncorrect. Therefore we tried to get the right answer by means of weights, preferences on the attributes and so on.

---

1 Nearest neighbour retrieval, with missing values being ignored, returns both CaseCorrect (100%) and CaseIncorrect (100%). Neither of the two other settings for handling missing values yields a higher similarity score for CaseCorrect than for CaseIncorrect.
Inductive Retrieval: The cluster tree contains the single decision node "Valve5Y1 is switched?". Since the new case does not specify this value, both the large and the small case are retrieved.

2 Using nearest neighbour matching with missing values being ignored, all other cases have a lower similarity score. Inductive retrieval returns 249 cases (because of the many missing values in the query case), leaving the user helpless.

When a tool could use a test selection component, we answered the given questions with the values of CaseCorrect if the question dealt with one of its attributes, otherwise with "unknown", thus simulating an end user who looked up additional fault symptoms on his CNC machine when requested for that by the CBR system.

Other possible features to improve the similarity measure were:

- Introducing different weights;
- Changing the handling of unknown or missing values;
- Combining inductive methods and nearest neighbour retrieval.

The effects of the different features are discussed below.

CBR EXPRESS As mentioned in the description of test T1, the weights for CBR EXPRESS were initially set to 1 for non-discriminant attributes and kept to 10 for all other attributes. Using these settings, CBR EXPRESS retrieved CaseIncorrect as, for example, PATDEX would do. The reason for this is that the computation of similarity is based on the percentage of the correctly-answered attributes of the *retrieved* case. This results in a score of about 43% (3/7) for CaseCorrect and 67% (2/3) for CaseIncorrect. Thus, CaseIncorrect takes advantage of the small number of questions answered in the query. If the user proceeds with the consultation by answering the correct values for "Valve5Y1" and "PipesClampingReleaseDevice", the similarity for CaseCorrect increases to 72%, which results in a correct answer. Besides answering more questions, the reaction of the system could easily be improved by introducing absence weights to punish the absence of an answer in the retrieved case. For example, a symmetric absence weight of -10 for all discriminant attributes will reduce the score of CaseIncorrect to 33% ((10+10-10) / 3) and, thus, result in the correct answer.

ESTEEM: With each of the similarity measures used, ESTEEM retrieved CaseCorrect and CaseIncorrect as the most similar cases. The changes in the similarity measures only improved the distance between these two cases and the other cases.

Default settings:

- CaseCorrect was retrieved with 4% similarity score, CaseIncorrect with 3% and 15 other cases with 1%. Amongst these 15 cases ESTEEM retrieved Case138.

Weights:

- All attributes got the weight 1 except the ErrorCode with 2. With this similarity measure ESTEEM retrieved CaseCorrect with 5%, CaseIncorrect with 4% and the same 15 cases as before with 1%.

Index built with the attribute ErrorCode:

- With this index, ESTEEM only retrieved cases with an exact match on the ErrorCode. These were CaseCorrect (4%) and CaseIncorrect (3%). Case138 was not retrieved because ErrorCode was different from the new case.

REMIND: Rather than demonstrating the effect each single integration step of background knowledge would have on the retrieval result, we describe how REMIND's features could be used to cope with this domain.

Causal rules like the one mentioned above could be included in the following way. Instead of the original attribute *IOStatusOUT24*, a new attribute was added to the case representation and was then considered in the clustering or matching processes. This new attribute's value was equal to that of *IOStatusOUT24* when this attribute had been specified by the user. Otherwise, it was determined by a formula implementing the causal rule.

The CNC MACHINE TOOL domain includes a diagnosis task. Instead of querying the system with a complete case description, the error symptoms had to be ascertained during a search procedure. This search procedure should be guided by the system by prompting the user for values of attributes considered to be relevant (test selection). Therefore, a consultation should begin with an inductive retrieval, where the system asks the user to enter values for unspecified attributes encountered during cluster tree traversal. Thus, a new case is created (or its initial description augmented) incrementally.

The order in which tests had to be performed could be highly relevant, since one test procedure might cause the impracticability of another test procedure. Although tools that solely rely on entropy measurement cannot take account of this, REMIND's concept of q-models can be used to influence the order in which attribute values should be determined.

Without a q-model, the following questions were asked during the consultation[1] with the new case (answers given by the user are shown in parentheses):

Questions:
ElectricalConnectionsToolGrip? (unknown)
ToolGrip? (unknown)
SpindleStop? (unknown)
ElectricalConnectionsClampingReleaseDevice? (unknown)
Relay21K7? (unknown)
Relay21K3? (unknown)
IOStatusIN35? (unknown)
IOStatusOUT24? (logical-0)
SafetyDiode5Y1? (unknown)

This resulted in the retrieval of 79 cases with 14 different diagnoses (retrieved under 8 clusters). Amongst them was CaseCorrect, but not CaseIncorrect because it was not indexed by "Valve5Y2 is not-switched". Thus, proceeding with a nearest neighbour

1 Bucket size for the construction of the cluster tree: 2; minimum number of cases to retrieve: 2.

retrieval on this set of cases determined CaseCorrect as the only case with 100% similarity score.

This process could still be improved in several ways: for example, including the given rule as described above prevented the system from prompting the user for the value of "IOStatusOUT24". In this way, the consultation process could be shortened. Nevertheless, the user had to answer a lot of questions that were irrelevant to the value specified for "ErrorCode".

An examination of the cluster tree revealed that "ErrorCode" was not chosen as the first attribute, probably because the number of cases stored in the case base were not sufficient to identify it as highly relevant for the "Diagnosis" value. One way to give REMIND this knowledge was to define a simple q-model that contained the information that the value of "Diagnosis" was directly influenced by the value of "ErrorCode".

A better way organising the cases would be the construction of a prototype for every possible value of "ErrorCode", because of the existence of different "error search schemes" for these values and the "error search schemes" could be used for the definition of q-models[1].

In our example, this partitioning of the case base caused CaseCorrect and CaseIncorrect to be contained in the same cluster, indexed by the prototype "ErrorCode is I59". Since our goal was to distinguish between those two cases, some work had still to be done. For some reason, REMIND's clustering algorithm refused to split clusters that only contain two cases (even if it could do so), so we have to create a third case (*e.g.* a copy of CaseIncorrect) before we could start clustering.

With these improvements, the consultation with the new case amounted to one question:

```
Question:
    Valve5Y1? (switched)
```

The only case retrieved was CaseCorrect.

S³-CASE It is possible to add a rule to the system similar to the one given in the question above. This rule is used for nearest neighbour matching to derive background knowledge. It is also possible to toggle, during consultation, to a system-driven consultation to ask for background knowledge to increase the information given by the current situation (new case). With the SMALLTALK-80 programming interface, it is possible to define attribute-specific similarity measures. Here the user can define unknown-specific similarity measures. It is also possible to use a table to define these measures. In combination with the user-definable weights, it is possible to refine and improve the measures to produce an improved solution.

---

1 This made it possible to disregard attributes for the values had been specified unnecessarily, i.e. their values were of no importance for the correct diagnosis.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ Does the tool employ rules or formulas to improve the retrieval process by deriving additional values? A causal rule could have the following form: IF ErrorCode = i59 & Valve5Y2 = not-switched THEN I/OStateOut24 = logical-1					
yes/no		●		●	●
○ Does the tool select the order of questions to narrow down to the right diagnosis?					
yes/no	●		●	●	●
○ Does the tool allow the handling of unknown values to be changed?					
yes/no	●[1]		●	●	●

Table 9.17. Correctness of the Execution System (Test T6)

9.2.6 Completeness of the Execution System

The two main points a system has to answer in relation to the completeness criteria are: Does the system cope with the complete domain? If not, is it able to detect that a conclusion is not possible? The following table shows that none of the tools answered both questions in a complete way.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ How easily can the execution system process and combine different kinds of knowledge to handle a domain as completely as possible?					
scale	1	2	2	4	4
○ By which technique is the execution system able to detect that a new case cannot, or probably cannot, be classified?					
knock-out attributes	●			●	●
thresholds	●	●			
background knowledge					●
induction			●		
prototypes				●	

Table 9.18. Completeness of the Execution System

9.2.7 Consistency of the Execution System

The consistency of the execution system is a major requirement in all applications. Obviously, consistency is not always compatible with other needs, such as the need for incremental operation: it is not possible to have an application system that evolves smoothly over time and that always provides the same output. Table 9.19 shows that all tools are consistent as long as their execution system is fixed.

---

[1] If an absence weight is introduced this penalises CaseIncorrect (33% instead of 65%) enough to let CaseCorrect win.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ Is the execution system stable over time, *i.e.* do you get the same result when using the same input data?					
scale	6	6	6	6	6
○ Where it is possible to determine the order of input, is the execution system sensitive to the order of input?					
yes	●				
not possible		●	●		
no				●[1]	●

Table 9.19. Consistency of the Execution System

TEST T7:

Repeating the Same Queries

We tested the behaviour of the execution system over time by applying the same queries twice. We used the TRAVEL AGENCY domain, taking all cases except 50 as reference cases and these other 50 as new cases. Each new case was presented to the tool twice to retrieve the most similar case with no changes to the execution system between the two retrievals. We note the percentage of identical results in the following table.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
% of identical results	100%	100%	100%	100%	100%

Table 9.20. Consistency of the Execution System (Test T7)

9.2.7 Effectiveness of the Execution System

The effectiveness of the execution system was measured by the number of interactions needed to reach a conclusion. We determined which systems optimise (minimises) this over time. Only those that incorporated index computation - KATE and S³-CASE - could perform such optimisation.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ Does the execution system optimise the number and/or the order of interactions over time based on its experience?					
yes/no			●		●[2]

Table 9.21. Effectiveness of the Execution System

1 During inductive retrieval it is not possible to change the order of input; nearest neighbour retrieval is not sensitive to the order of input.

9.3 Development System

In this section we cover the behaviour of the CBR tools during application development - the process of using the generic platform in order to build an execution system that runs a particular application. We investigate the ability to handle noisy or incomplete data during this process and consider issues such as performance, consistency, effectiveness and adaptability to user's needs.

9.3.1 Ability to Handle Noisy Data During Application Development

Tools applying nearest neighbour matching for similarity assessment are inherently resilient to a certain kind of noise. The effect of noisy data is more-or-less compensated for by computing the similarity. A second possibility for handling noisy data is to use data types that define partially unknown or uncertain values such as symbol taxonomies, or symbol ordering. The existence of such data types enables the user to represent domain definitions that can - with some restrictions - handle noisy data. The effect on classification accuracy caused by noisy data during application development is generally much higher than that caused by noisy data during consultation.

TEST T8:

Systematic Introduction of Noise during Application Development

In this test, we introduced noise into the case data in the same way as in test T1. This time however, we kept the new cases unchanged and changed the reference cases instead. For each of the four steps of this test, we used the whole CAR domain case base with its 205 cases, respectively changing 10%, 20%, 40% and 60% of the attribute values of every case. With these case bases we retrieved the most similar case for 50 new cases taken from the 205 original cases and computed the percentage of correctly classified cases for each degree of noise.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ How does the system support the development of an execution system that is resilient to noise?					
tolerance interval		●			
linearly changing distance	●	●			
symbol taxonomy			●	●	●
symbol ordering				●	●
local distance measures					●
○ TEST T8					
10% noise	100%	100%	88%	92%	100%
20% noise	98%	100%	86%	84%	98%
40% noise	96%	98%	54%	44%	94%
60% noise	80%	94%	56%	44%	84%

Table 9.22. Handling Noisy Data During Application Development (Test T8)

9.3.2 Ability to Handle Incomplete Data During Application Development

TEST T9:

Systematic Reduction of Available Information within the Reference Cases during Application Development

This test is the counterpart of Test T2. We took the 205 cases of the CAR domain and deleted 10%, 20%, 40% and 60% of the attribute values respectively, thus obtaining four case bases containing different amounts of information. 50 of the still-complete cases were taken as new cases for each of the four case bases. Again we computed the percentage of correct classifications for each percentage of information that was removed from the reference case data.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ Is it possible to represent unknown values explicitly in the execution system?					
yes/no		●	●	●	●
○ Test T9					
10% reduced	100%	100%	100%	100%	100%
20% reduced	100%	100%	100%	100%	100%
40% reduced	100%	96%	98%	100%	100%
60% reduced	100%	94%	94%	100%	98%

Table 9.23. Handling Incomplete Data During Application Development (Test T9)

KATE: In contrast to test T2, the effect of missing values for KATE is very low. Since all values of attributes on which decisions have to be made during tree traversal are specified in the new case, it is not necessary to follow more than one path. Thus, the reduction of information can only cause bad classifications if leaf nodes contain more than one diagnosis. In the test domain, this occurred very seldom because, in general, a few attributes were sufficient to identify the class value. The reduction of information necessary for a classification is the fundamental idea of a decision tree built using an information gain measure.

9.3.3 Performance of the Development System

Note that the times required by the tools are not directly comparable since they accomplish tasks of different complexities. For example, REMIND proceeds linearly, searching the case base for nearest neighbours and, thus, probably builds up no particular indexing scheme, whilst KATE constructs a decision tree based on the best information gain.

TEST 10

Building Speed for Automatic Application Development

This test measures the time required by the CBR tools to build execution systems with differently-sized case bases. The case base used in this test was the whole TRAVEL AGENCY domain with its 1,470 cases and subsets of 200, 400 and 800 cases. Since neither CBR EXPRESS, nor ESTEEM have explicit options to build an indexing mechanism (decision tree, *k*-d tree, *etc.*), this test was not directly relevant for them. Results of explicit generation of the indexes are given in Table 9.24a. For REMIND, we were not able to build a cluster tree on the whole database in a reasonable time. Since REMIND builds binary trees, this problem is probably due to the high number of values for some attributes (in addition, there appears to have been memory problems with this version that have been solved in a newer release). The building time for KATE corresponds to the induction tree building and for S³-CASE, to the *k*-d tree building.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ TEST T10 (time in seconds)					
building time for 200 cases			41[1]		18
building time for 400 cases			93		36
building time for 800 cases			195		104
building time for 1,470 cases			376	X	1654[2]

Table 9.24a. Building Speed of the Development System (Test T10)

Since the results of section 9.2.4 (see Table 9.13, speed of the execution system) clearly show that the retrieval time in CBR EXPRESS is not affected by the number of cases in the database, we assumed that some indexes were built automatically. If there was no index at all, the retrieval time should be linear with the number of cases. There is no indication in the manual about an indexing mechanism. We assumed that if an indexing mechanism was built automatically, this must take place during loading and we thus measured loading time for the same case bases as in the previous table. The values thus correspond to memory management activity following data loading and possibly to the building of an implicit indexing structure.

1 Supplier's note: this refers to the time for constructing a static decision tree for KATE-INDUCTION. For KATE-CBR, a path in the decision tree is dynamically built according to the user's query. With KATE 4.0, which has been rewritten in C, AcknoSoft claims to have obtained generation of the whole tree for 1,470 cases in less than 7 seconds for the travel domain.

2 The long time required for 1,470 cases results from the garbage collection invoked by the Smalltalk-80 system.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ TEST T10					
loading time for 200 cases	73	23	16	17	83
loading time for 400 cases	127	38	32	21	117
loading time for 800 cases	241	67	64.	75	239
loading time for 1,470 cases	445	120	118	181	408

Table 9.24b. Time Required to Load the Case Base and Possibly Build Hidden Indexes (Test T10)

9.3.4 Consistency of the Development System

The development system should give consistent results when building an application. Given the same input on two different occasions, the development system should produce the same execution system. All tools except REMIND were consistent with respect to this requirement.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ Is the development system independent of the order of input?					
yes/no	●	●	●	1	●

Table 9.25. Consistency of the Development System

9.3.5 Effectiveness of the Development System

Many parameters have an influence on the quality of the development system. These include cost functions, system optimisation over time, input data minimisation, *etc.* Table 9.26 summarises these.

1 For nearest neighbour retrieval, REMIND performs a linear search over the case base. The order in which retrieved cases with the same similarity score are presented reflects the order in which they were stored (*e.g.* imported or created) into the case base. As mentioned in a comment to test T1, a restriction on the number of nearest neighbours to be retrieved can thus introduce a random element into the retrieval result.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ In what way(s) does the development system help minimise the amount of input case data necessary to cope with the intended application domain, *e.g.* by use of background knowledge and/or learning?					
default values			●	●	●
learning of weights					●
background knowledge				●	●
case testing	●				
○ In what way(s) does the development system make effective use of additional parameters?					
size of a final leaf			1	●	●
entropy			●		●
global sim. measures	●		1	●	●
local sim. measures			1		●
inter quartile distance					●
q-model				●	
○ Does the development system support the optimisation of the execution system over time based on the system's experience?					
yes/no					

Table 9.26. Effectiveness of the Development System

9.3.6 Adaptability of the Development System

The last section of this chapter deals with the adaptability of the development system, *i.e.* the possibility of modifying its output according to its input.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ Is the development system able to explicitly generalise data?					
yes/no			●	●	
○ Is the development system able to automatically purge data?					
yes/no					

Table 9.27. Adaptability of the Development System

1 Supplier's note: available in KATE 4.0.

Chapter 10: Evaluation based on Ergonomic Criteria

Ergonomic criteria include those dealing with application development, explainability and modelling support, user acceptance of the CBR tools and the organisational impact of the underlying technology. These criteria complement the technical criteria. In this chapter we cover ergonomic criteria and give a summary of our impressions of the user interfaces, gained whilst using the CBR tools during the evaluation process.

10.1 Application Development

The ergonomics of an application development interface are very important if we want to use a tool to develop an application. We investigated three kinds of criteria: the control of the application development, the possibility of validating the resulting system and the ease of acquiring and maintaining new data.

10.1.1 Control of Application Development

The following questions are concerned with the ability of the tools to represent background knowledge of the domain, whether they allow the inclusion of different knowledge modules into the current consultation, and the effect on the quality of the execution system. The more knowledge a system is able to integrate into its reasoning processes, the more likely it is that changes in the behaviour of the execution system will be observed.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S3-CASE
○ To what degree is the application development adaptable to special needs?					
scale				4	4
○ Can a domain expert control the complexity of the execution system?					
scale	1	3	3	4	4
○ What is the effect on the quality of the execution system?					
scale	1	2	3	5	5

Table 10.1. Application Development

10.1.2 Validating and Testing the Execution System

It is an important feature for a CBR tool to be able to test its output. The developer is then able to correct the system before deploying it and to test whether the overall application is understandable. In other words, does it use knowledge that is directly understandable to ease validation and acceptance?

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ Does the development system provide any automatic or manual facilities to test and validate the resultant execution system with respect to the input data and additional test data?					
scale	2		4	4	4

Table 10.2. Validating and Testing the Execution System

10.1.3 Acquiring and Maintaining Knowledge and Data

Incremental operation is the characteristic most frequently accorded to CBR tools. Tools should therefore be flexible with respect to knowledge updating, data acquisition, revision, *etc.* The following table shows that there are promising approaches to these problems, but that they are not yet entirely solved.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ To what extent does the system cope with domains that need frequent updates?					
scale	5[1]	1	3	5[2]	3
○ Does the application development support automatic long-term optimisation of the execution system?					
scale			2		2

Table 10.3. Acquiring and Maintaining Knowledge and Data

10.2 Explainability and Modelling Support

As stated in section 10.1.2 (Validating and Testing the System), the user's acceptance of the technology is greatly increased when the tool provides self-generated explanations of its features and/or its conclusions and when it is well documented. For the application developer, an ideal tool should have a very good user manual; context-sensitive on-line help; it should be able to backtrack over its decisions for better comprehension of the mechanisms it used to reach them; and it should have facilities for knowledge description and utilisation. Most of the tools performed relatively well on these various points.

The next section provides a deeper look at end-user acceptance.

---

1 CBR EXPRESS provides a number of reports that are automatically generated for keeping track of the usage of the application system. Adding new cases or questions causes no problems and it is possible to store them in different storage classes (*e.g.* to distinguish between *resolved* and *unresolved* cases).

2 Templates can be used to retrieve cases that are subject to changes. Attributes can be added or deleted at any time.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ Is the execution system able to justify and explain its conclusions, decisions, questions *etc.*?					
scale			2	4	
○ Is the system well documented?					
scale	4	2[1]	3[2]	6	
○ Does the tool employ an on-line-help function?					
yes/no	●	●	[3]	●	
○ How well does the system support domain modelling?					
scale	3	3	1	5	5

Table 10.4. Explainability and Modelling Support

10.3 User Acceptance

Dissemination of CBR technology begins with the user's conviction that it can be of use. After technical validity, user acceptance is a key point in selling a product.

10.1 User Interface

The first aspect of user acceptance is the user interface; the user can be the application developer or the final user of an application. The quality of the interface must be srongly emphasised. Below we summarise the main qualities of the interfaces and in the next sub-section expand on this for each tool.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ Is it possible to configure the execution system's interface according to the current user's experience and needs?					
scale	4		2	4	2
○ Does the user interface support different kinds of user?					
yes/no	●	●		●	
○ Does the consultation follow the logical flow expected by a domain expert from his field of expertise, *i.e.* is the system comprehensible to him?					
scale	3		3	4	3
○ Does the user interface improve the user's motivation and thereby improve the system's acceptance?					
scale	4	2	3	4	3
○ Does the user interface support the use of additional multi-media?					
graphics	●	●	●		●
hypertext	●		●		●
sound	●	[4]	●		●
video	●		●		●

Table 10.5. User Interface

[1] Supplier's note: ESTEEM 1.4 has new documentation which describes many areas more clearly.
[2] Supplier's note: KATE 4.0 has complete documentation about the tool and the DLLs.
[3] Supplier's note: KATE 4.0 does.

10.3.2 Summary of the Interface

In this sub-section we summarise how to use some basic features of the user interface for each tool. The descriptions include how to define a case base and a similarity measure, how to input cases and background knowledge, how to use such knowledge, how to retrieve and adapt memorised cases and how to combine different schemes of reasoning. In addition, we include some screen-shots that help explain certain points.

CBR EXPRESS: CBR EXPRESS is a one-window application that can easily be controlled by drop-down menus. According to the different CBR tasks, the user can choose different panels to enter data. The number of available menu items and panels varies from maintenance to user mode. In most cases, the user can either select a menu item, press a button, or use control codes to get the same effect. Due to the unity of all panels, the overall system can be used intuitively. In addition, on-line-help is available on every feature.

The case base is defined in maintenance mode using the case, the question and the action panel. The question panel allows the definition of attributes and the related types. In addition, the user can enter descriptive text and can control the weights for each question. Cases are normally entered using the case panel by selecting the appropriate questions and actions (target attributes) from a list. The question selection and ordering are controlled with buttons. After all questions have been answered, a case can be tested against the current case base and, if not redundant, can be stored.

Since the similarity measure is mainly influenced by attribute-specific weights, it is controlled using the question panel. Besides this, it is possible to control general weights using a special case-base-options window.

After building the application, the whole system is switched to user mode, restricting the available panels to the search and tracking panel. The first step in handling a request would normally be to enter the customer data into the tracking panel. The upper part of the panel is, quite simply, an interface to the customer database. The lower part is reserved for call tracking. After entering the customer data, the operator can switch directly to the search panel for retrieval.

The search panel contains three main fields. The first is used to enter a textual description that will start the search. The second then shows questions to be answered to narrow the search. An arbitrary question can be selected for answering via a "pop-up" window that shows the possible answers. The system will react to the answer by updating the current set of matching cases shown in the bottom field. The cases displayed there can be browsed using buttons. If one solution is promising, the "end search" button brings the user back to the tracking panel to where the retrieved information is copied.

---

[4] Supplier's note: ESTEEM 1.4 can make use of *.wav files, *.avi files, etc. in an application and in the user interface.

All additional functions such as import/export features, report printing and preference settings are accessed using common-style drop-down menus. It should be mentioned again that the predefined look of the interface could be adapted to special needs if TOOLBOOK is available.

Although most panels have a well-designed look-and-feel, the search panel could be better arranged. If the cases contain more than only a few slots, it becomes hard to compare two cases because they are not presented next to each other. The retrieved case can only be inspected by scrolling, which is even harder due to the different order of questions.

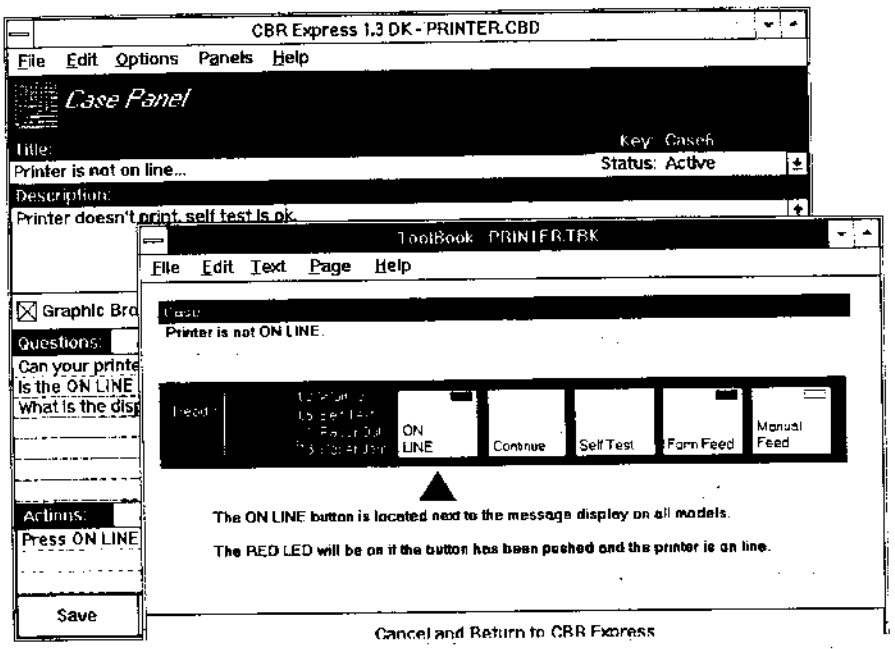

Figure 10.1. Adding pictures in CBR EXPRESS

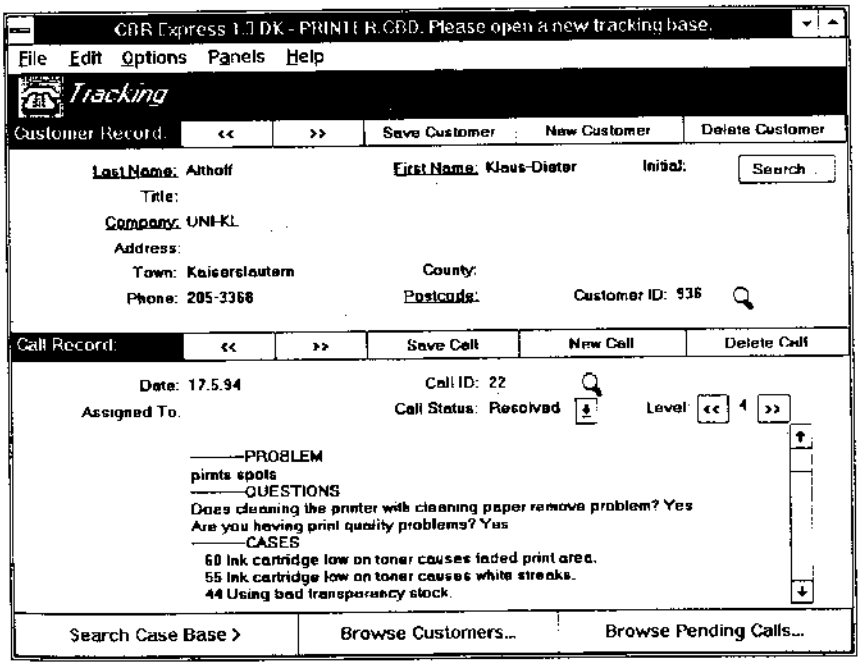

Figure 10.2. Help-desk System of CBR EXPRESS

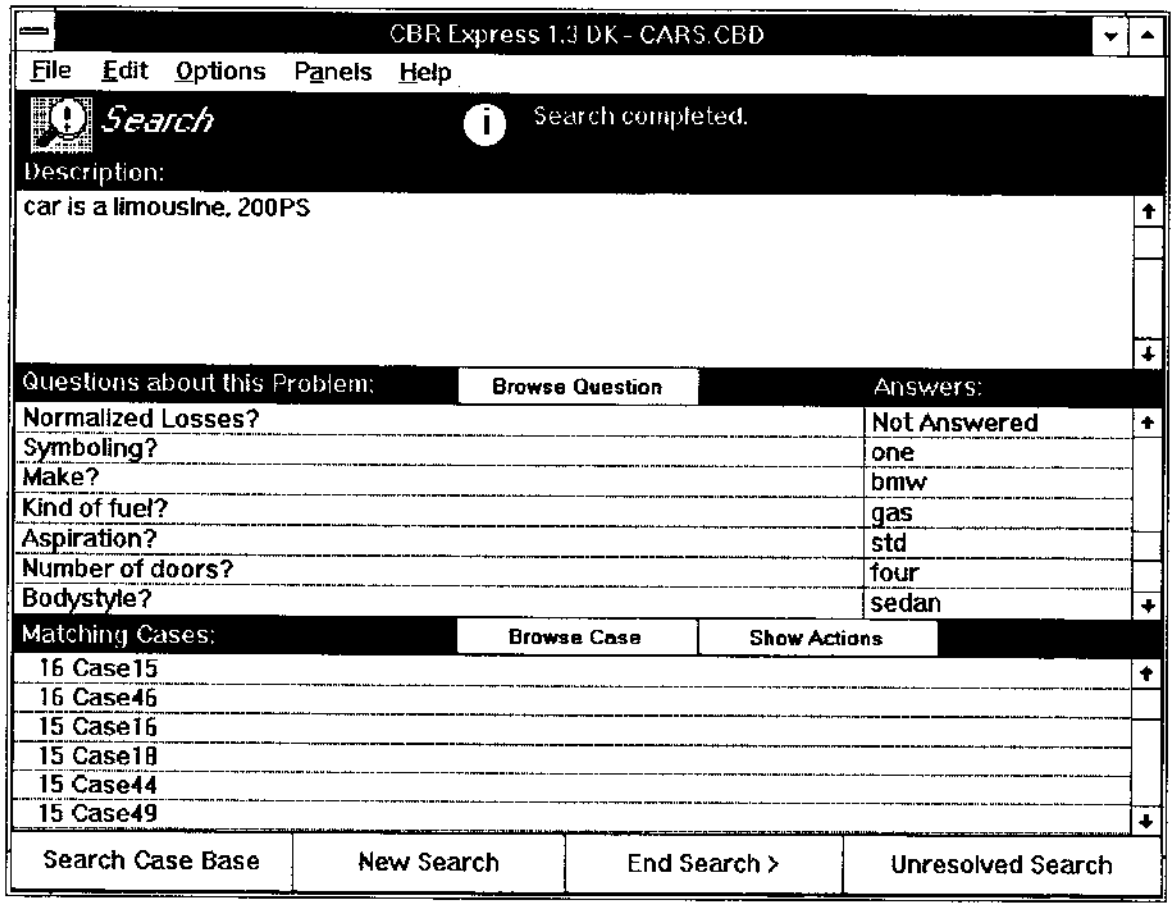

Figure 10.3. Cases Retrieval in CBR EXPRESS

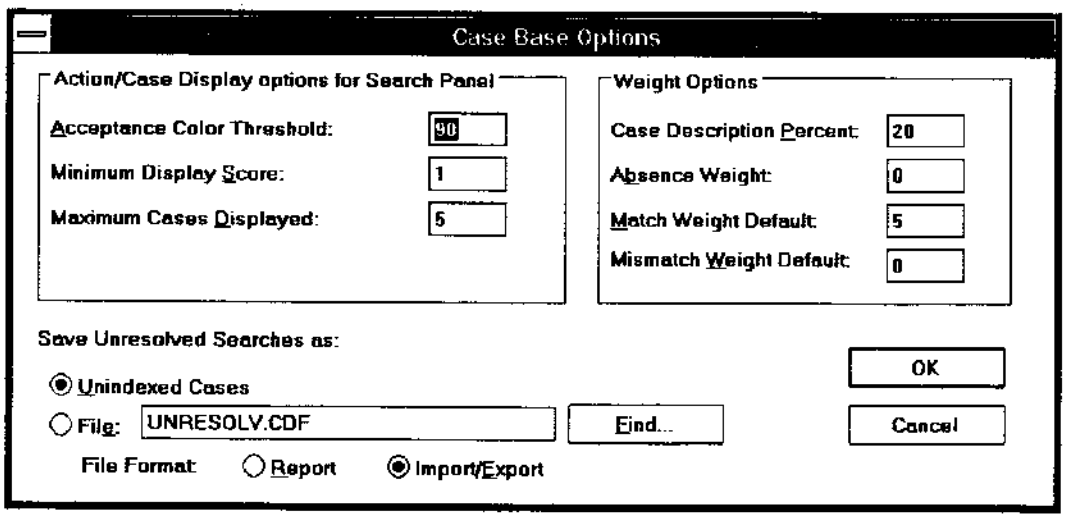

Figure 10.4. Similarity Definition in CBR EXPRESS

ESTEEM: The main window of ESTEEM's user interface includes an icon for each of the main data objects of an application: case bases, cases, similarity definitions, rules and rule bases. Two further icons are available to define and to run the end user interface, *i.e.* the execution system. Other features such as file operations or the indexing facility can be accessed from a menu bar.

When the user has clicked on an icon, he has to decide whether to create, select, edit or delete an object. This mechanism is a little unwieldy, but it is necessary since ESTEEM can handle several case base definitions and similarity definitions, although only one of each can be active at a time. For this reason, each of these objects has to be given a unique name.

In the case base definition editor, the developer can enter the names and types of the case attributes. When choosing a numeric or symbol ("one of a list") type, he is prompted to enter the allowed values for these attributes in a separate window. All or part of the case base can be saved in a database. This allows for database files to exist on a server and to be used by multiple clients - using the database engine as the mechanism for managing concurrence.

The similarity definition editor consists of a headline which contains global parameters and one line for each attribute in the case structure. The global parameters are the threshold for the minimum similarity score for a retrieved case and the weighting method. The available weighting methods are "Feature Counting" (no weighting), "Weighted Feature Computation" (weights are explicitly given) and "Inferred Feature Computation" (weights can be computed by rules). The lines for each attribute contain a check box to determine whether the corresponding attribute takes part in the similarity assessment, the local similarity measure to be used and the weight of the attribute. Again, the developer may be prompted for additional parameters such as the tolerance interval for numeric comparisons. If a local similarity measure or a weight should be computed by rules, the user enters "Inferred" and the name of the rule base to be used.

The case editor offers an input field for each attribute of a case. For symbol and Boolean attributes the user can obtain a list of the possible values from which he can choose one with the mouse.

ESTEEM's rule editor shows two fields in which the user enters the antecedent and the consequence of an IF-THEN rule. These parts of a rule are expressed in the Kappa Application Language (KAL), in which ESTEEM itself is written. The use of KAL makes the rules a very powerful instrument of ESTEEM because they are able to call any KAL function provided with Kappa-PC or loaded from a separate file. On the other hand, this makes the use of rules difficult for people who are not familiar with Kappa-PC.

All rules that take part in the computation of a value are collected in a rule base. The rule base editor is just a text window in which the names of the selected rules are entered.

As a last step of application development, the user has to define the end user interface. In the corresponding editor, he has to choose three subsets of the case attributes. The first one defines the attributes that can be entered for the case retrieval. The second set contains up to two slots which are displayed as a brief description of the retrieved cases. The attributes that can be viewed when selecting one of the retrieved cases are collected in the third set.

In a further panel of the end user interface editor, the developer can decide whether the end user should be able to modify the case base or the similarity definition. Here he can also select a rule base for automatic adaptation as well as some KAL functions that will be executed in special situations of the consultation process.

The ESTEEM end-user facility itself is entered by selecting the corresponding icon in the development system or by launching it separately from the Windows program manager.

It consists of a window with a menu bar and some buttons to trigger functions and the target case window. The user can enter his new case into the target case window and press the "Retrieve" button. The retrieved cases are then shown in a window that displays the similarity score, the name of the case and the attributes selected in the end-user interface editor. The user is then able to inspect one of these cases by selecting it with the mouse. He may have the option to change the selected case manually and store it in the case base, or have it adapted automatically, depending on the settings.

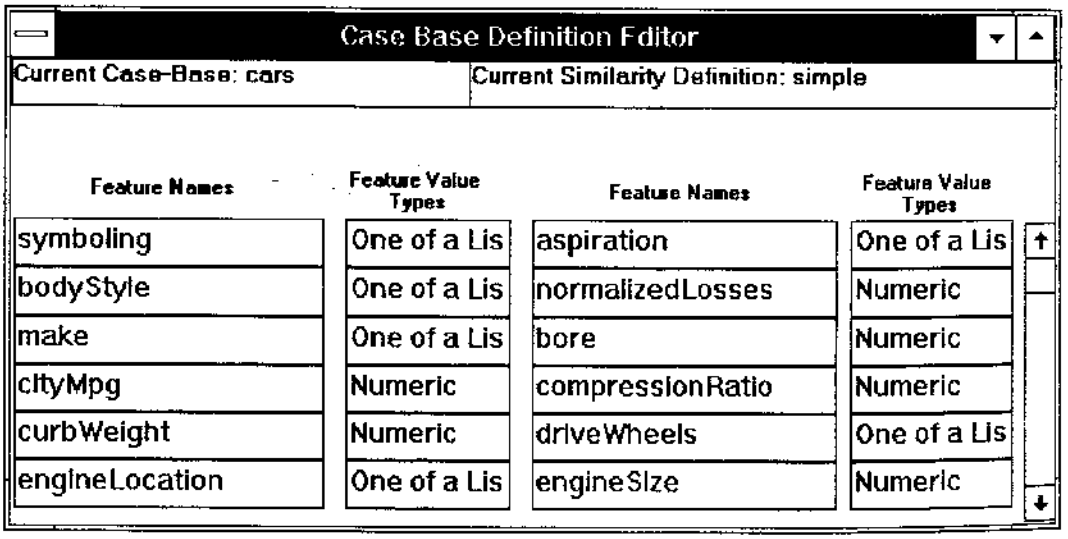

Figure 10.5. Case Base Definition Editor in ESTEEM

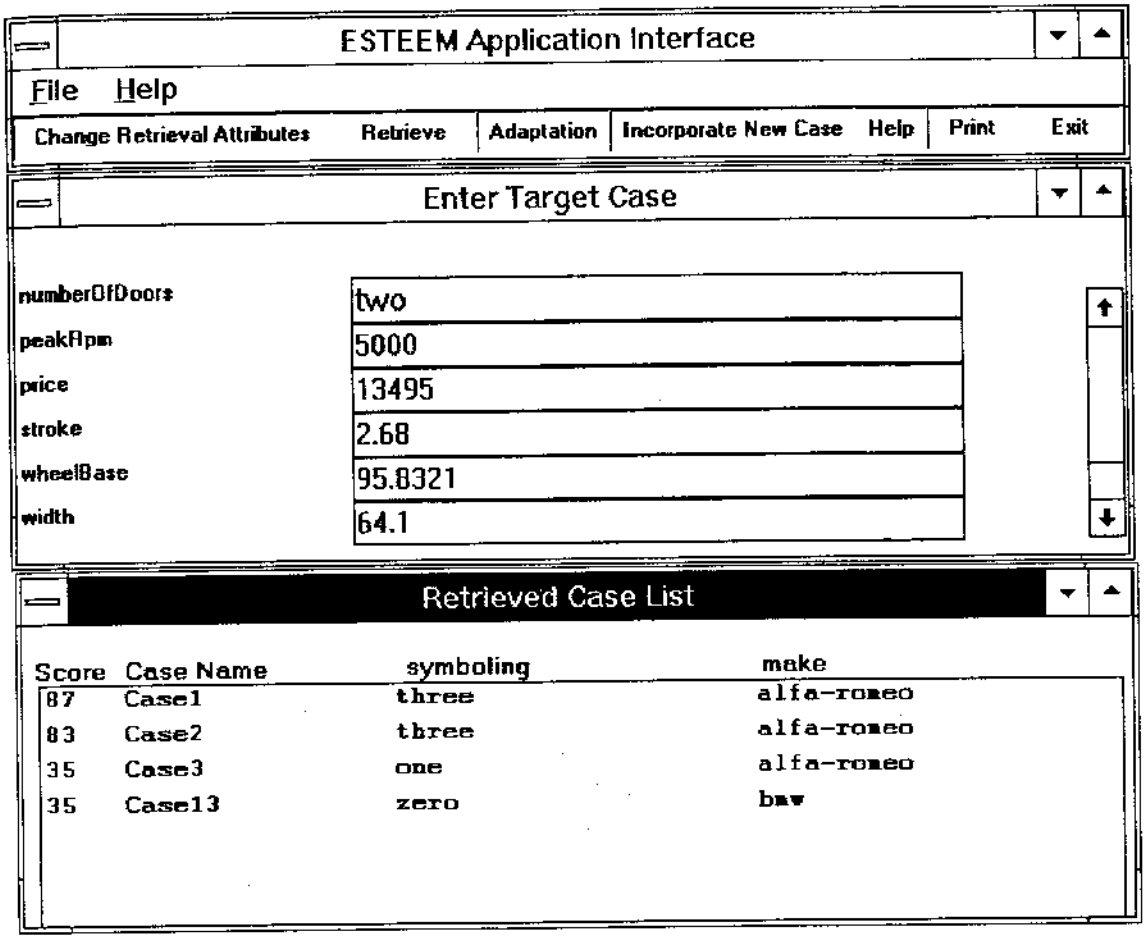

Figure 10.6. Cases Retrieval in ESTEEM

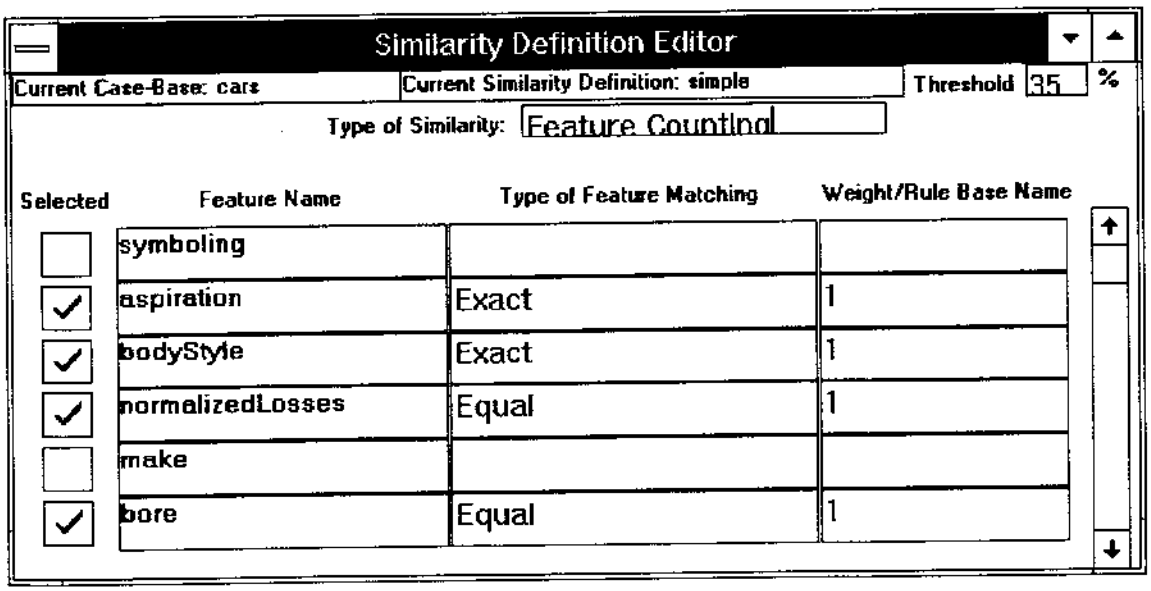

Figure 10.7. Similarity Definition in ESTEEM

KATE: In this evaluation we used the KATE 3.0 tools running on a PC under Windows 3.1. The main window consists of several pull-down menus. Integrated into the system is an editor to manipulate the source files containing the domain specifications. It is possible to define a domain description and to compile the description into the system in CASUEL syntax, or in an EXCEL-compatible format. These formats are described in the user's manual and by various examples delivered with the tool. The user also has the option to use an external text editor in which to define the domain description. Both descriptions include the definition of the domain theory and the case base. After the domain has been defined and the case base is compiled into the system, the domain definition can be visualised in a graphical window.

To run the KATE-INDUCTION component, it is necessary to create a decision tree. During the construction of such a tree, information relating to the process is displayed in a text window[1]. The resulting tree can be archived to disk in a runtime format. There is a graphical browser to inspect the tree. It is possible for a domain expert to modify sub-trees manually. It is also possible to view the cases attached to a node and to support the users in getting information about the tests in the tree. From this tree definition, the user can generate short-cut rules to improve the consultation of the tree. The construction of this set of rules is viewed in a text window.

Using the KATE-CBR component of the KATE toolbox, computing the decision tree before using the execution system is not necessary. Here the whole case base is dynamically indexed, based on the user's input, to lead to a conclusion. The next best question is calculated by the attribute with the best information gain with respect to the new case. In this mode the user can choose between different targets for consulting the case base..

During a consultation of the decision tree, the user is asked for different values of attributes to lead him to a conclusion. The consultation of the decision tree is completely menu-driven and is constrained by the domain definition. The result of this consultation is a diagnosis, a probabilistic estimation for different diagnoses, or a refutation if the system was not able to classify the user's input. The referenced cases of the conclusion can be inspected with a graphical interface. We were not able to evaluate the nearest neighbour retrieval module of KATE 4.0 nor the object model editor and questionnaire of KATE-EDITOR since they were not provided to us in time for the evaluation.

---

1 Supplier's note: this feature has been removed from KATE 4.0 since the tree building is too fast for efficiently displaying the information during the construction process. The information is written directly into a text file that can be read by the user afterwards.

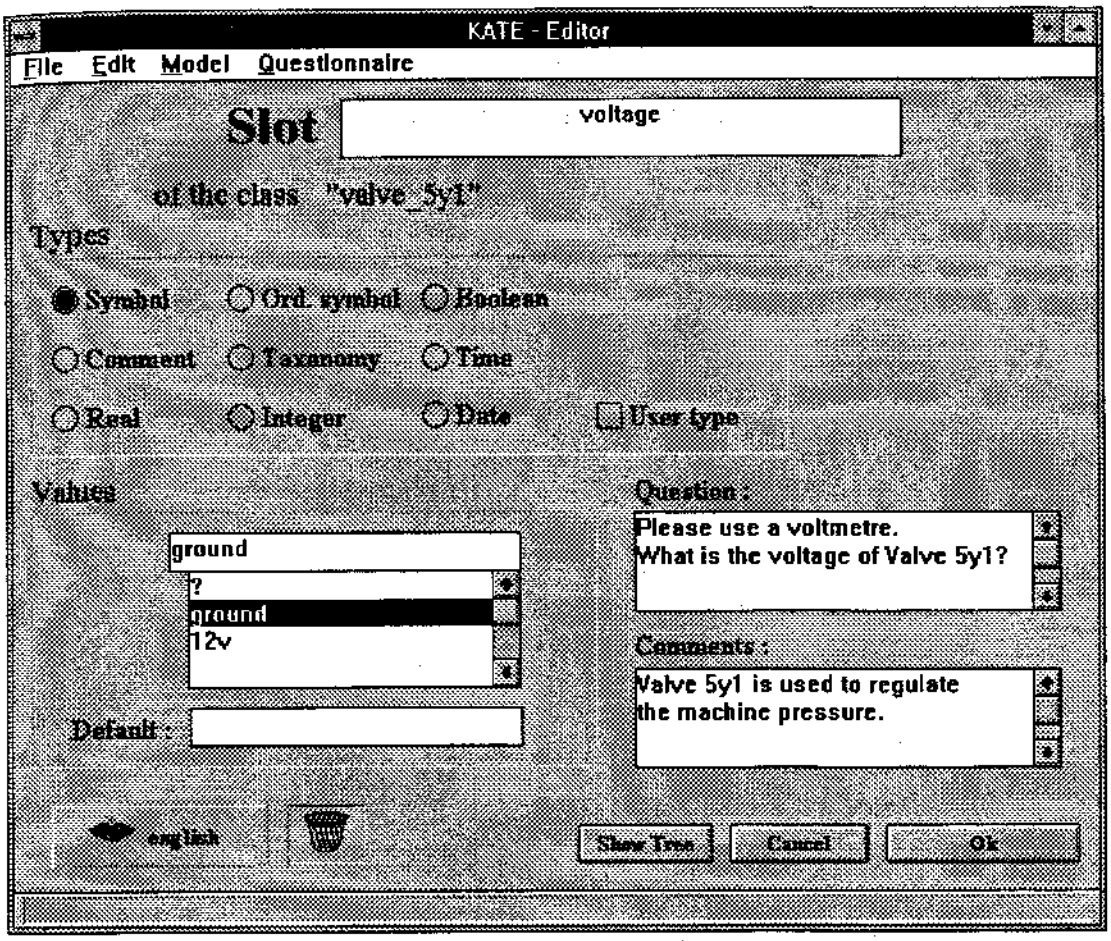

Figure 10.8 KATE-EDITOR offers an Object-Oriented Knowledge Editor

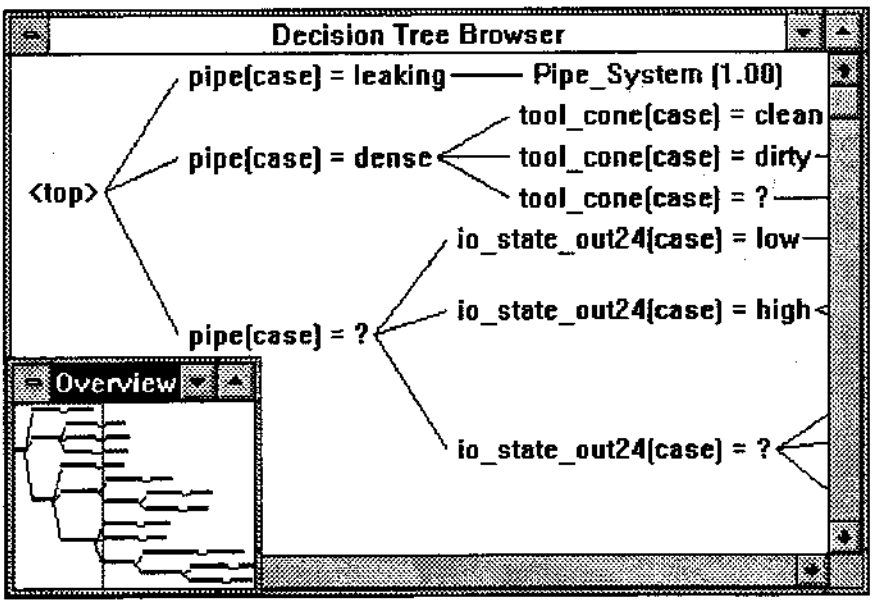

Figure 10.9. KATE-INDUCTION Decision Tree Browser

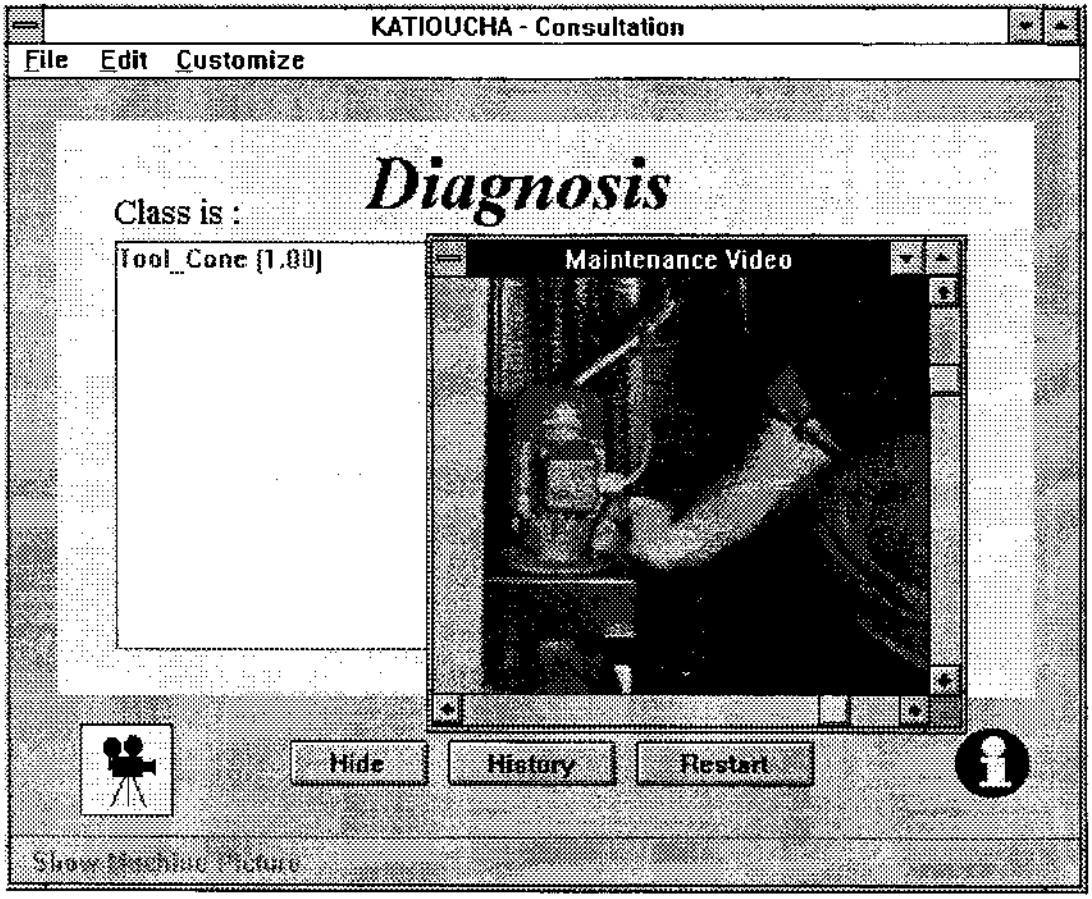

Figure 10.10. KATE-RUNTIME can include pictures and videos

REMIND: The REMIND development system consists of nine different graphic-oriented editors, especially designed for maintaining the different components of a case base. We will briefly describe seven of them in the order in which they would usually be used to build a new case base, leaving aside data import and the form editor.

• The Attribute Editor

The attribute editor is used to define or modify case attributes. To create a new attribute, the user is prompted for it's name and type. All further attribute-specific information (*e.g.* range restrictions, default value or the handling of missing values in prototypes) can be specified or modified later using the editor. Attributes can be created, renamed or deleted any time.

• The Symbol Editor

In addition to the attribute types built-in to REMIND, it is possible to define a symbol taxonomy. More specifically, it is possible to define a certain kind of graph with labelled nodes (the labels being the symbol values) and four kinds of edges between two nodes: "parent-of" (or the inverse: "child-of") and "less-than" (or the inverse: "greater-than"). Figure 10.11 shows a simple example. The label nodes *any colour*, *white*,

yellow, *etc.*, all represent possible symbol values. The links with no arrow-head represent "parent-of" links and links with an arrow-head represent "less-than" links, *e.g.* ordering colours according to lightness. Every user-defined node has one or more parent node allowing symbol aliasing.

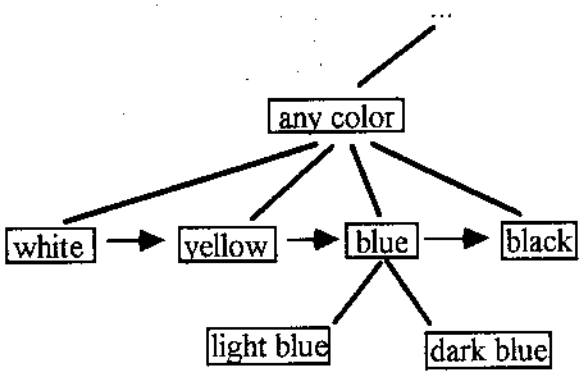

Figure 10.11. Symbols in REMIND

An attribute of type *Symbol* has to be associated with one of the nodes as its root node, being the point of entry into the taxonomy to restrict the set of possible values. Only labels of nodes that can be reached by following "parent-of" edges starting from that root node are valid values for the attribute. Given the taxonomy in the example above, an attribute specifying *any colour* as its root, would have as its set of possible values *white, yellow, blue, light blue, dark blue* and *black.*

The taxonomy and ordering information are used for both the clustering process and the computation of the similarity between symbolic values during nearest neighbour matching. In the example above, *white* will be considered to be more similar to *yellow* than *blue*. Also, *light blue* will be more similar to *blue* than *black.*

The symbol editor displays the case base's symbol taxonomy in almost the same way as the figure does, so working with the editor is much like working with a drawing-program. The user selects a node and drags it over another symbol to establish an order link, or creates a new "child" via a menu command, *etc.* The symbol taxonomy can be edited at any time.

• The Formula Editor

REMIND offers a very large set of built-in functions, from trigonometric up to text-parsing functions (too large, in fact, to give an exhaustive overview here). From these functions, the user is able to define formulas, using a succession of operations and conditional expressions as the only control structures (*i.e.* no loops can be defined). Such formulas can be associated with case attributes to determine their values automatically. Since several of the built-in functions could only be defined by using loop-structures, it is difficult to formally describe the set of expressible functions. The result type of a function can be any of fourteen types.

The formula editor permits the graphical definition and editing of a formula associated with an attribute by constructing a data-flow graph. Again, working with this editor is much like working with a drawing-program, choosing "tiles" representing the basic functions from a menu, placing them on a workspace and connecting them to define the flow of data. This process is supported by the possibility of selecting a sample case and of displaying intermediate results of the computation at each graph node.

The formula editor is used for the definition of both functional dependencies between the attributes of a case and adaptation formulas to be associated with selected attributes.

• The Importance Editor

The importance editor is used to define different weights' vectors and to choose a matching method for missing values (to be used for nearest neighbour retrievals), attach importance settings (to be used for cluster tree generation), or to identify attributes for which adaptation formulas will be defined.

A user might define several weight vectors in advance and will select one from this set when starting a nearest neighbour retrieval. For the weight vector definition, every attribute will have one of the following three settings: 1) an exact match is required (knock-out attribute), 2) the attribute will be ignored in the computation of the similarity score, or 3) the attribute has an integer between 0 and 99 attached to it as its weight.

Clustering requires the selection of one attribute containing the case's class value (*i.e.* the target attribute) and the specification of which attributes could have an effect on the class value.

• The Cluster Editor

Rather than clustering the whole case base uniformly, it might be necessary to use different importance settings, *etc.*, for certain subsets of cases - for example, because certain attribute values define specific contexts. For this reason, REMIND offers the possibility of defining a hierarchy of prototypes and allows the system to start the clustering process in each prototype individually.

The cluster editor gives a graphical representation of the current cluster tree, which allows the user to delete unwanted splits, edit the prototype hierarchy, examine mixed clusters, *etc.*

• The Q-Model Editor

Clustering can be performed with or without a q-model (qualitative model). By building a qualitative model, it is possible to describe qualitatively the effect of a change in one attribute's value on that of other attribute values. The q-model defines a precedence ordering on the set of attributes: an attribute *A* will only be considered as a candidate on which to base a split (*i.e.* to construct a decision node with) if all attributes whose

values are (directly or indirectly) influenced by *A* have already been used as split attributes in the path from the root to the cluster to be split.

There are three kinds of influence that an attribute *A* might have on another attribute *B*:

i) *B* increases when *A* increases and decreases when *A* decreases.

ii) *B* increases when *A* decreases and decreases when *A* increases.

iii) *A* influences *B* in an unspecified way.

The q-model is depicted as an oriented acyclic graph, the attributes representing nodes with three different types of link according to the kinds of influence mentioned above. In addition to the attributes already defined using the attribute editor, the q-model also allows the creation of "virtual" attributes, that combine one or more case attributes into a new attribute. Virtual attributes become useful if one is unable to exactly specify a functional dependency between these attributes, yet still wish to enrich the case representation to improve indexing. Links of the kind (i) and (ii) pointing to a virtual attribute will be used to compute a system-internal value to be stored in that attribute, that can be used for the construction of decision nodes in the cluster tree.

• The Case Editor

Cases can be created, deleted or edited using the case editor. It provides all the functionality to maintain the case base. In addition, one of the three retrieval methods can be invoked from here. The set of cases retrieved can be browsed in a "case-comparison" window, which is basically just another case editor, functionally extended to depict the new case together with a retrieved case for ease of comparison, and to allow adaptation to be started using the currently displayed retrieved case. Unlike a case's formula attribute, whose value is updated any time according to the values the formula depends on, adaptation formulas will only be invoked on request. Given a (partly specified) new case and a retrieved case, the user can selectively invoke adaptation formulas associated with attributes of the new case whose value has been left unspecified. The resulting value will then be stored into the new case attribute.

For nearest neighbour retrieval, the user selects a weights' vector and specifies the number of cases to retrieve. The case-comparison window presents the cases retrieved together with their similarity score. For an inductive retrieval, the user specifies the minimum number of cases to retrieve and whether the system should prompt her/him during tree traversal for attribute values that have not been specified in the new case but are required to determine the path to be followed in the decision tree. At the end of the retrieval process, REMIND not only opens a case-comparison window but also an "explanation" window that contains information about how many cases were retrieved, under how many clusters, what class values occurred how often among these cases, which prototypes had been used and the list of splits encountered.

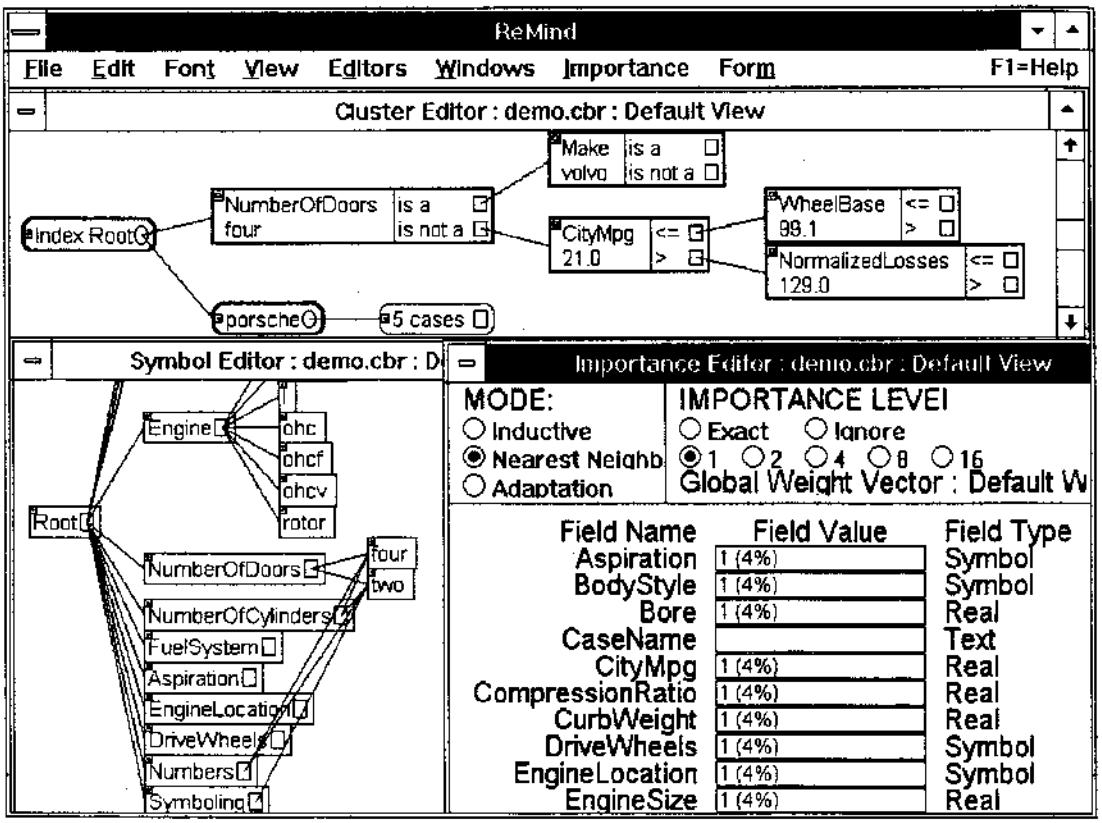

Figure 10.12. Cluster Editor, Symbol Editor and Importance Editor in REMIND

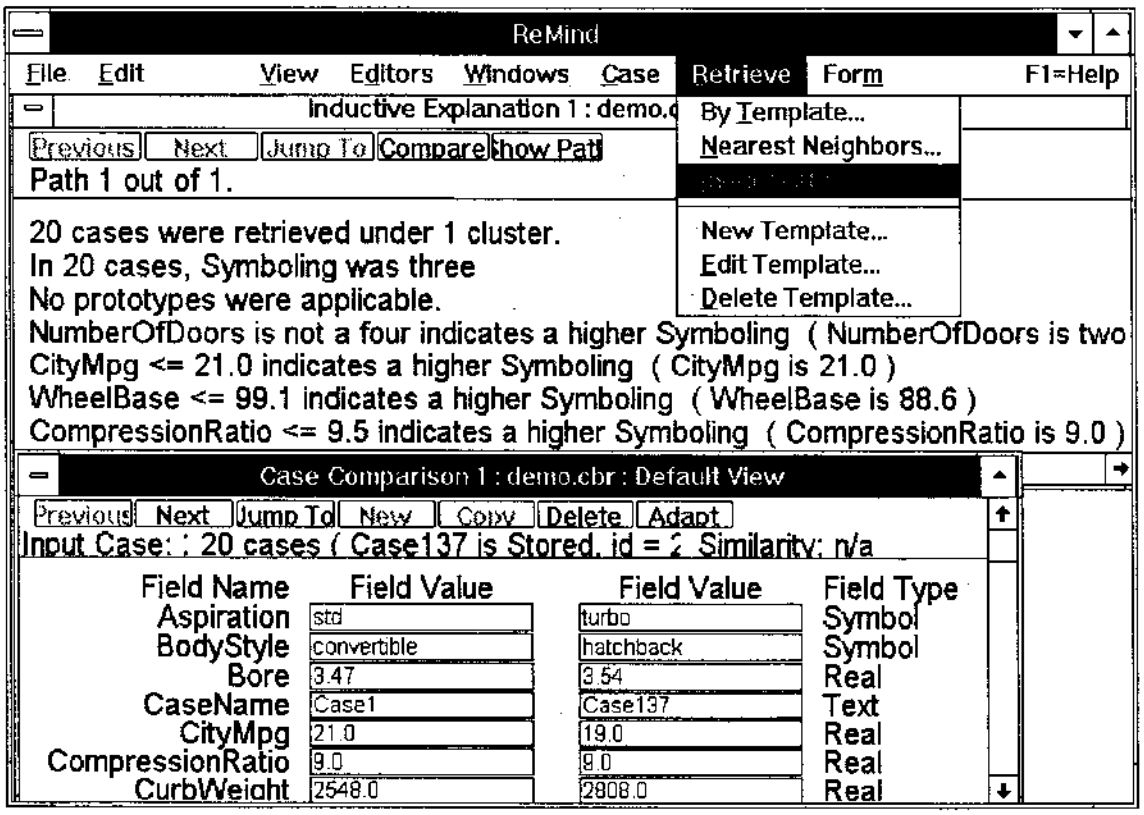

Figure 10.13. Cases Retrieval and Explanations in REMIND

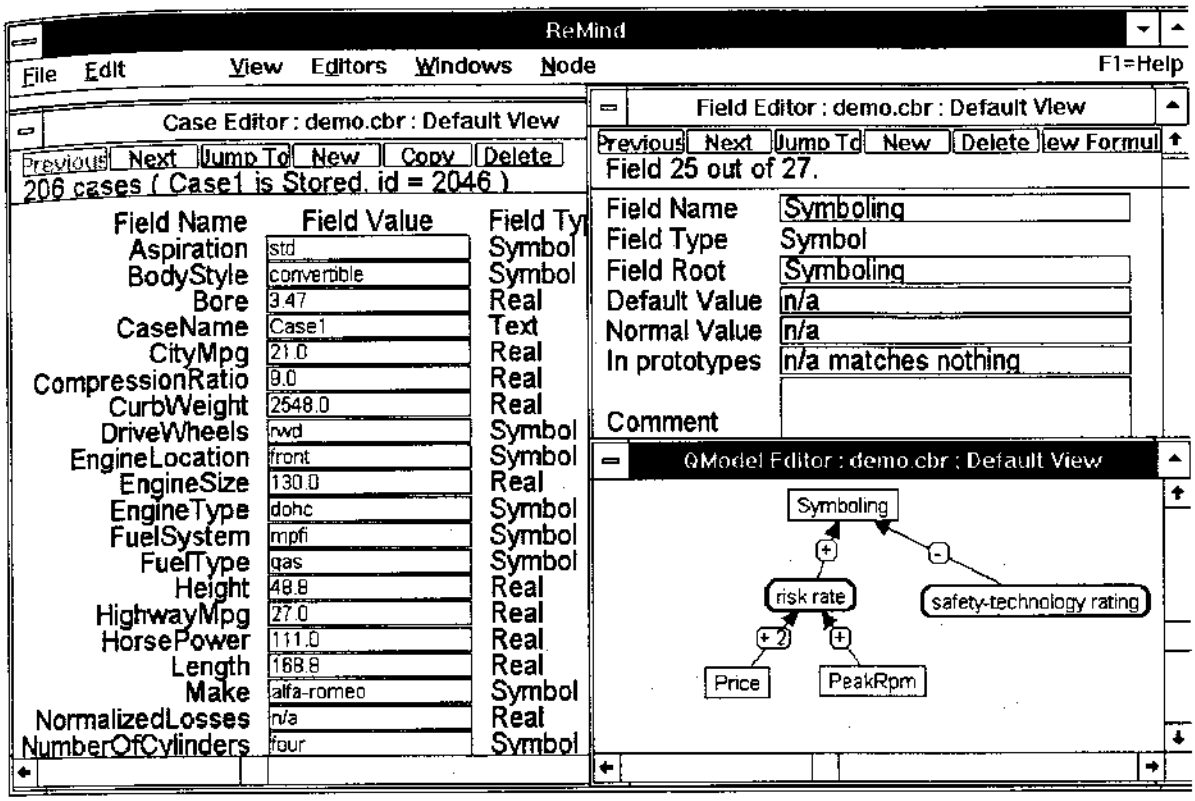

Figure 10.14. Case Editor and Q-Model Editor in REMIND

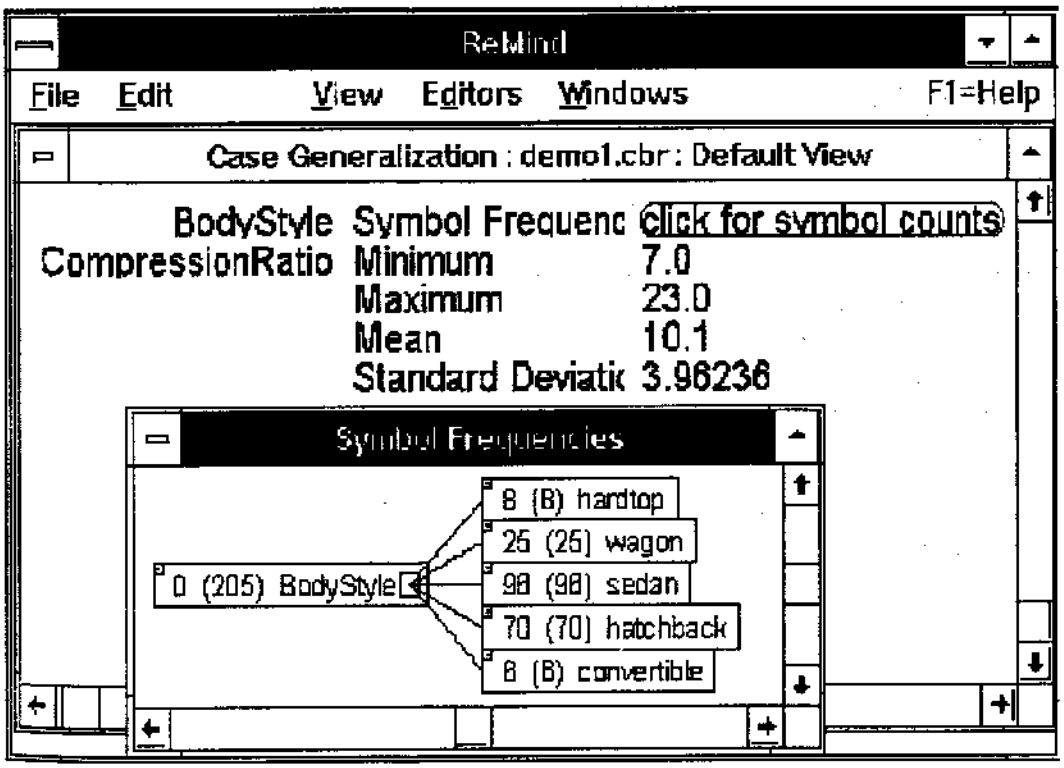

Figure 10.15. Data Generalisation in REMIND

S³-CASE: After starting the S³-CASE system, the user gets a "launcher", which can be described as a permanent pop-up menu. From this launcher, there are several sub-menus available. In the following, these sub-menus are briefly described on an abstract level.

• The Type Editor

In the type editor, the user has the possibility of refining the predefined types given in S³-CASE. The standard types are: Boolean, Integer, Real, Symbol, Ordered Symbol, Sound, Graphics, Video, String, Text and Symbol Taxonomy. The types Integer, Real, Symbol and String can be refined by a sub-range definition, or an enumeration of the allowed values for the types. In addition, for Ordered Symbols and Symbol Taxonomies, it is possible to define a special order or hierarchy for the refined types. The type hierarchy can be visualised in a textual list or a graphical browser. It is possible to define a type name in different languages to make the description understandable for foreign-language-speaking users. The type definitions or the graphically visualised type hierarchy can be archived in a postscript file that may be printed out with a postscript-compatible printer. The type editor checks the correct refinement of super-types, such as correct sub-definitions of sub-ranges. All these types are used in the concept editor to declare the attributes' types. The type editor offers the opportunity to define type-specific similarity measures with a SMALLTALK interface or in a table format to define a similarity between each value for a given type. The type descriptions are compiled into the SMALLTALK-80 image so that they may be accessed from the program interface for other similarity measures or rules in other interfaces using SMALLTALK-80. These similarity measures are managed by a similarity library.

• The Concept Editor

The concept editor is used to define concepts: *i.e.* sets of different attributes. Concepts are linked into a concept hierarchy. It is therefore possible to represent structured domains with a hierarchical concept description of a case in the defined domain. The concept hierarchy can also be visualised in a textual list or a graphical browser. A postscript file can also be generated to print the current concept hierarchy. It is also possible to define concepts in different languages and to design comments for them in a hypertext interface. In the concept editor, the user defines for every concept a set of attributes and their types. It is possible to define a default value and an attribute-specific question for every attribute in a concept. These attribute types can be combined in different conjunctions, disjunctions or intervals. The user can distinguish between local and global or discriminant and non-discriminant attributes. As in the type editor, there is a SMALLTALK-80 interface to define concept-specific similarity measures.

• The Domain Editor

Here the user can define a target concept and a target attribute of the concept hierarchy. It is also possible to declare the name of the domain and the connected file names of the CASUEL description. The default language for the S³-CASE system is determined here.

• The Case Base

In the Case Base Editor, the user can enter new cases, modify existing cases or destroy old cases. To reuse entered data, it is possible to adapt new cases from old cases. It is also possible to print cases in postscript. The user can choose between two different editors to enter new data about the case or to edit an existing case. Different languages and automatic counting of cases are supported.

• The CBR Shell

Case base indexing can be organised linearly, by a *k*-d tree, or value-indexed. The generation of the index can be influenced by many parameters for the special needs and wishes of the user. It is possible to choose between different splitting modes, similarity measures, bucket sizes, case selections, test selections and memory cache sizes. To represent the functional dependencies between case attributes, the user can define rules of attribute constraints for different concepts. The system may also learn weights for the attributes or they may be manually designed by a domain expert. The indexing scheme can be viewed with a graphical browser. To test the effectiveness of the current indexing scheme, the user can consult the case base directly from the CBR Shell.

• The Utilities Menu

In the utilities menu, the user can load a CASUEL description of a domain and a case base into the system. It is also possible to export a CASUEL description of the user-designed domain and case base for data transfer to other systems or to archive the knowledge of the domain.

• The Consultation

In the consultation window, the user can enter a new case to calculate the nearest neighbour cases of the case base. During a consultation of the case base the user can freely toggle between the system-driven and the user-driven consultation modes. In the first mode, the system asks for the next value of the attribute that will most quickly retrieve a case with high similarity. In the second mode, the user can freely choose the order of input data and can calculate the most similar cases. From here the user has an overview of the indexing scheme using a graphical browser. There is also a bar-chart overview of the calculated similarity value of the ten most similar cases.

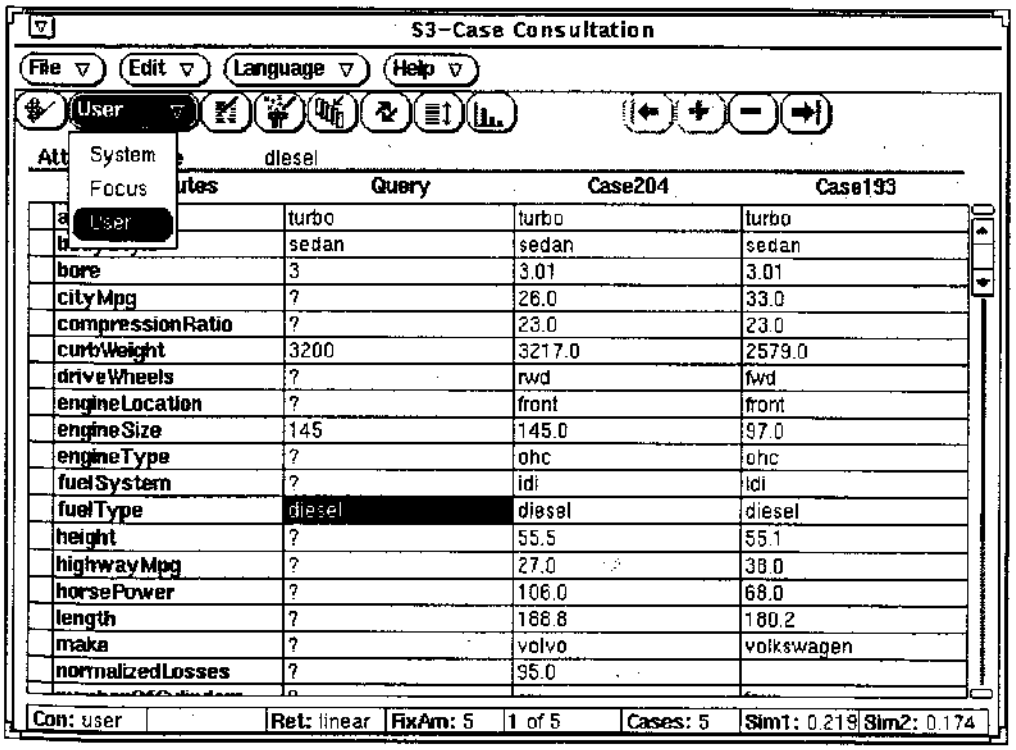

Figure 10.16. Case Retrieval and Comparison in S³-CASE

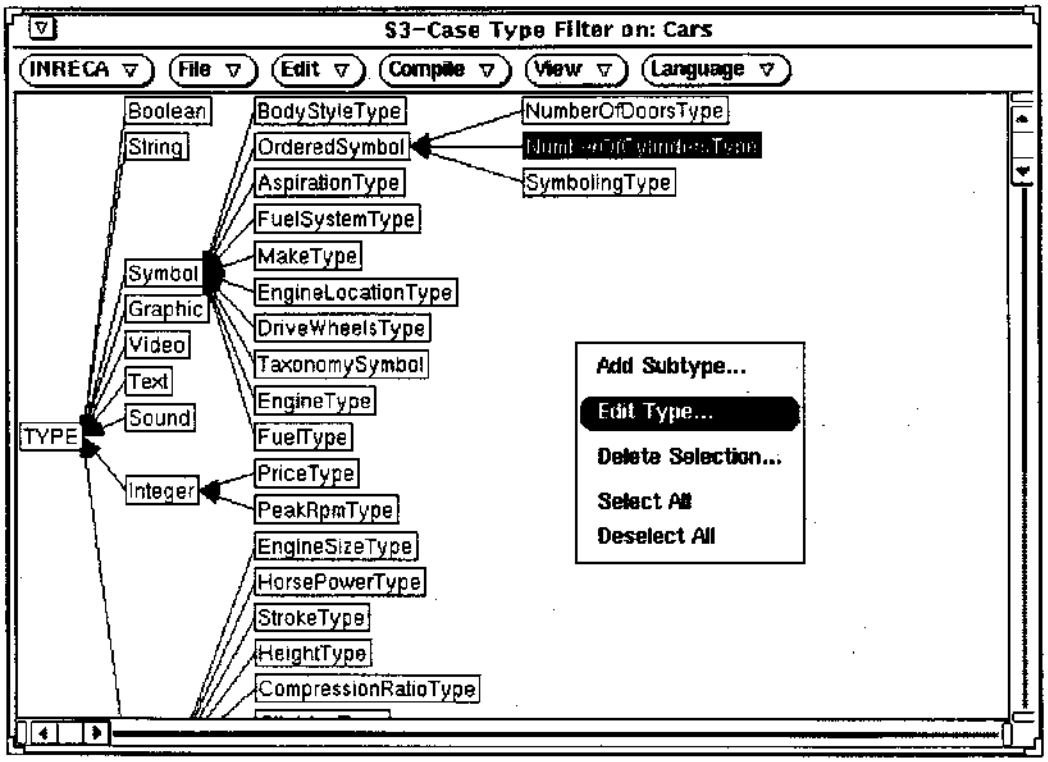

Figure 10.17. Symbol Editor in S³-CASE

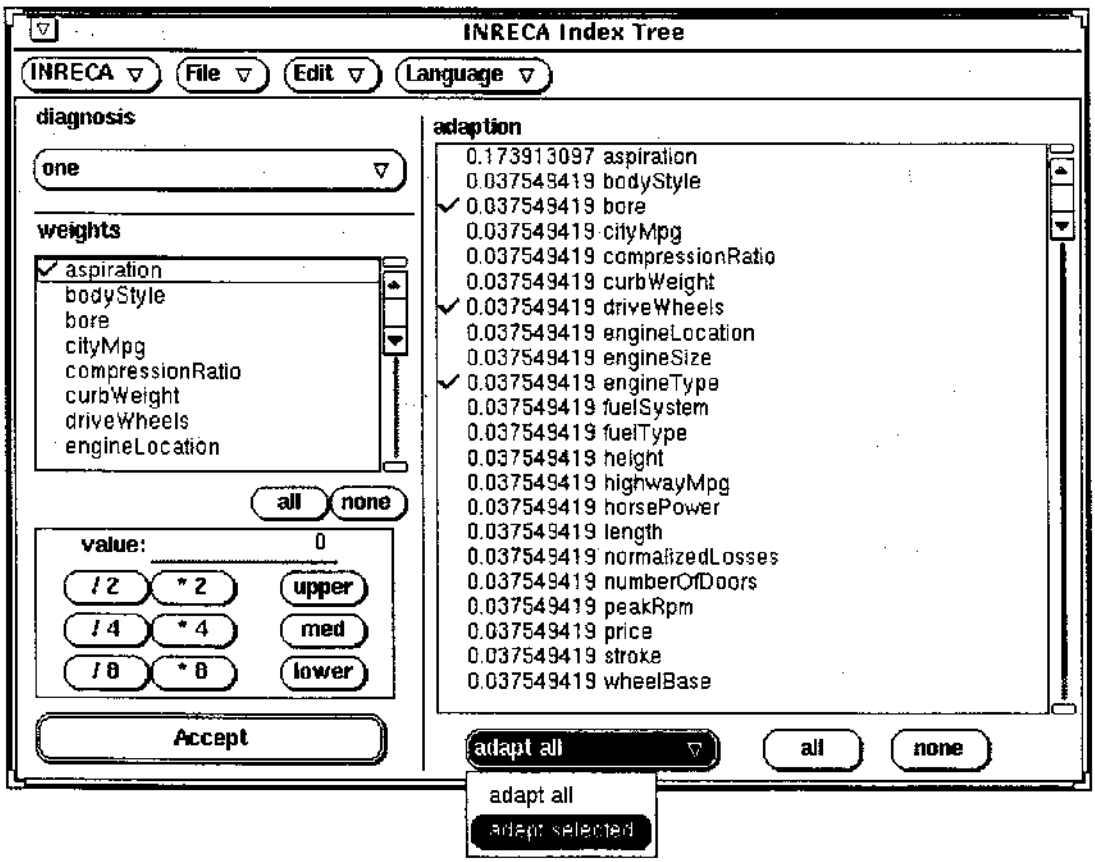

Figure 10.18. Importance Weights Editor in S³-CASE

10.4 Organisational Impact of the Technology

CBR technology has a double impact on the daily work of a company. Firstly, it can appeal to - or frighten! - domain experts, by enabling them to describe their expertise. CBR can be used to capture daily experience, as at Lockheed in the *Clavier* system (Chapter 6). Secondly, the "naive" end-user is also involved in his work when using a help-desk tool. It is therefore important to offer an easy-to-learn user interface.

	CBR EXPRESS	ESTEEM	KATE 3.0	REMIND	S³-CASE
○ Is training of a domain expert necessary?					
scale	1	3	4	3	3
○ Is training of a user necessary?					
scale	1	1	1	1	1
○ Is the execution system a tool to capture experience to prevent a loss of know how?					
scale	4[1]	3	3	4[2]	5

Table 10.6. Organisational Impact of the Technology

10.5 Interface with the Outside World

The last section of this chapter deals with case base import/export facilities, communication with other programs and the user interface.

CBR EXPRESS: CBR EXPRESS provides import and export utilities to transform a case base into a readable ASCII file. The purpose of these utilities is mainly for porting, merging and editing tasks. We used this facility to import all the test data. Due to the high number of keywords and the good formatting, exported case bases can easily be adapted with an ordinary editor. On the other hand, transforming data from another format to CBR EXPRESS import format is slightly more complicated than with other tools. To help with this, a nearly complete and correct BNF listing comes together with the handbook. Unfortunately, the number of cases that can be imported in one transfer is restricted to a few hundred.

CBR EXPRESS can import files from Oracle, Paradox, dBase, Sybase, Informix, Ingres and DB2. Using the Dynamic Data Exchange protocol from the Windows environment, CBR EXPRESS can be driven from a client Windows application. Depending on this application, CBR EXPRESS might be running entirely in the background to answer the clients' requests. The handbook gives several concrete examples for DDE requests. The user interface can be edited using Asymetrix' TOOLBOOK and the database can be put onto a network to support multi-user access with record locking and unlocking. The

[1] Concentrates on the documentation of the expert's knowledge.
[2] Concentrates on the operationalisation of the expert's knowledge.

supplier strongly emphasises its natural language interface capability (especially for the Compaq application). However, it is unclear how much of this had to be programmed manually and is tied to that particular application.

ESTEEM: ESTEEM is able to load and save all parts of an application, *i.e.* case base definitions, cases, similarity definitions, rules and rule bases in separate ASCII files. These files can be edited manually or used for data exchange with other applications. However, only the format for case files is described in the documentation.

Case data can also be read from Lotus 1-2-3, dBase and SQL databases. The new version also supports, according to the supplier, MicroSoft's Access. Lotus 1-2-3 and dBase files can also be written by ESTEEM.

The Windows DDE interface is fully supported by ESTEEM. This means that ESTEEM can be driven from another Windows application. In the same way, ESTEEM can control other programs and exchange data with them. The system does not support shared databases on a local network.

KATE: KATE is available as a set of DLL's that can be integrated in an existing application to add inductive and CBR features. For loading a domain into the system, it is possible to read a CASUEL description or ASCII descriptions in an EXCEL compatible format. The use of third-party databases is also supported (Oracle, Paradox, dBase, Sybase, Informix, Ingres, DB2 *etc.*) and so is Microsoft's ODBC (Open DataBase Connectivity). Databases can be exported and the system offers OLE links to EXCEL to graph similarities and to monitor trends in the case base. The user interface can be edited using Asymetrix's TOOLBOOK. The database can be put onto a network and supports multi-user access with record locking and unlocking. An extension is being developed to support heterogeneous networks with different platforms.

REMIND: Instead of importing data files in some fixed format, REMIND permits the graphical definition of data-flow graphs, similar to the formulas mentioned above, to express functional dependencies between attributes. Since the rich set of primitives from which these graphs can be built include conditional expressions and basic text-parsing functions, REMIND can cope with files containing more complex case descriptions, *e.g.* in a limited subset of the CASUEL syntax[1]. Files containing the case data in record format can easily be imported by specifying the delimiting characters (or fixed positions within a record) and a simple one-node graph for each attribute.

By appropriately extending the import functions associated with the attributes, it is possible to keep track of the symbolic values occurring in the cases being imported and automatically insert them into the symbol taxonomy, probably sparing the user a lot of typing.

Sadly, REMIND does not come with a set of predefined import functions to read databases or export files created by any of the common database programs. What is

---

[1] For this, all the attributes have to be defined and, of course, the case representation has to be flat.

worse is the complete absence of any facility to export data from a case base, so that other programs could process them. Apparently, it is not even possible to transfer a case file between versions of REMIND on different platforms. This also implies the impossibility of merging two case bases, which would be necessary for a distributed development of case libraries and useful if several people are using a case base at different places, storing new cases during their work.

Cognitive Systems offers a "C Library of Functions" for integrating REMIND functionality into CBR applications. Using these functions is the only way to extract data from a REMIND case base. Stand-alone case-based applications can be deployed via runtime copies of REMIND. The system does not support shared case libraries on a network in an easy way but, apparently, it is possible to program this by hand using the C libraries.

S³-CASE: The system has an interface to import or export a description of the whole domain and the case base in CASUEL Syntax. In addition, the system supports DDE and RPC communication. The directly-supported database systems are: Oracle, Sybase, Objectivity, GemStone and ArtBase. S³-CASE supports multi-user access via a network through these databases. The user interface can be edited using SMALLTALK.

Chapter 11: Applications Developed with the Tools

Disclaimer: We asked all tool vendors to supply two references of successful applications of their systems. Where possible, we checked the references and were pleasantly surprised to find that the tool suppliers had not greatly exaggerated reality. Most customers seemed happy with the tool and with the relationship they had developed with the tool vendor. They all stated that availability of first class consultancy, in addition to the tool, was a key factor in the successful implementation of CBR technology at their site. Nonetheless, the material presented below is mostly based on the information supplied to us by the vendors and we assume no responsibility for its accuracy.

11.1 CBR EXPRESS

11.1.1 Application I - Help Desk for American Airlines' SABRE software system

In this application CBR EXPRESS had to be integrated into a large help desk system for SABRE Agency Data Systems. This help desk was already in use by over 80 people. The use of the system produced a reduction of 30% in the time to resolve customer issues and a 70% increase in the number of problems resolved on the first call. A broad range of topics with voluminous documentation had to be covered. To fulfil these requirements CBR EXPRESS as a tool for problem resolution was combined with third-party products for call tracking and indexed text retrieval. After being successfully deployed, CBR EXPRESS showed the following benefits: it provided a learning aid for employees with limited domain knowledge; it provided mechanisms for policy management; and it allowed a single point of access to extensive documentation through case annotations. Return on investment was less than one year.

11.1.2 Application II - Service support system for Compaq

The second application serves as an example for call avoidance by the use of CasePoint (the runtime-version of CBR EXPRESS). Compaq expanded its support from dealer to direct customer support, after having identified service as a key market differentiator. Complex customer inquiries relating to operating systems, networks, GUIs and applications needed to be answered. CBR EXPRESS was used to build a system (called SMART), now deployed at service centres world-wide. The case base authoring was performed by support analysts over a four month period and the system is still evolving. CBR EXPRESS showed multiple benefits such as a 74% improvement in first call solutions. Besides saving time and resources it enables analysts to spend more time on difficult and unusual problems. Due to this successful experience, Compaq decided to distribute support directly to the user. Quicksource, another application delivered by Compaq, provided direct support for Pagemarq laser printer users. It was built using CasePoint and resulted in a 20% reduction of calls.

Besides these two examples CBR EXPRESS is used by companies such as Microsoft, IBM, Dun and Bradstreet Software and Black & Decker to distribute troubleshooting knowledge. The following payoffs can be seen as typical (according to Inference):

- Cost reduction up to 40%;
- 30% reduction in the time to resolve customer issues;
- 70% increase in the number of problems resolved on first calls;
- 50% reduction of training time for new support analysts.

These pay-offs result in a return on investment of less than a year.

11.2 ESTEEM

11.2.1 Application I - Cost and sales prediction for SHAI

A good example to demonstrate ESTEEM's abilities is an application implemented at SHAI, Inc., an architectural/engineering (A/E) company in California. The task of this application was to predict the cost of an A/E project by retrieving the most similar one from a case base of past projects. Beyond the required technical data of an A/E project, the case structure also contained information about the client, including a classification (federal, commercial, private, *etc.*). The first prototype of the system contained about three hundred cases, covering a period of five years. After two weeks of tests, the similarity measure was improved and another twenty years' cases were entered to build the final system. More details about this application can be found in [Richard H. Stottler, CBR for Cost and Sales Prediction, AI Expert, August 1994].

A reduced version of this application (reduced case structure with a few invented cases) is included in the collection of sample applications shipped with ESTEEM).

11.2 Application II - Process planning support for NASA

ESTEEM was used to develop techniques for planning Space Shuttle processing for NASA, Kennedy Space Centre. This included a Case Base of past Shuttle Processing plans. Many of the cases were plans to be executed in special circumstances, such as an unusual trip for an orbiter to the Palmdale maintenance centre in California. Experienced mission planners were interviewed to identify plan structure, relevant planning techniques, heuristics and data. This knowledge was captured using a combination of rules and object-oriented representations. Plans were represented as a hierarchy of intelligent resources with referenced resources, which were also intelligent entities. Rule bases could be attached to these entities to allow very complex plans to be built from relatively simple components. Adding new types of plan and new types of resources can be performed easily by non-computer literate personnel. To plan for the current circumstances, past similar plans are retrieved and modified automatically. These automatic modifications can be overridden by the users in a number of ways, most simply by editing the generated plans.

Planning techniques were developed to automate the multi-mission planning process. The Phase I effort resulted in a successful scheduling prototype and a complete design for an automated manifest planner. The two-year Phase II effort resulted in the implementation of the full-scale automated manifest planner and development of feasible techniques for automated entry and learning of planning knowledge.

11.3 KATE

11.3.1 Application I - Troubleshooting the BOEING 737 jet engines

A decision support system was developed on behalf of Cfm International for troubleshooting the CFM-56-3 aircraft engines of the BOEING 737. The database contains over twenty-three thousand cases and is added to by some fifty events per day. We used a combination of induction and CBR and inductive technologies in order to speed up case retrieval time. These were reduced to less than one second. The scope of the application is to minimise downtime of the engines, thus reducing unavailability of the aircraft, and to decrease the number of wrong diagnoses that may lead to taking an engine off the airline. The time required for diagnosis represents roughly fifty percent of downtime (the remaining time is used for repair) and the objective of the system is to have the time required for diagnosis divided by two. Three problems have been considered:

- Updating the troubleshooting manual from observed faults: this is particularly interesting for new engines in order to rapidly share practical troubleshooting experience (transfer of expertise). This is critical for equipment that remains in operation over a long period of time and where experience is likely to disappear when specialists retire or change jobs;
- Help-desk system: the after-sales support service at Cfm International uses the tool to support the airlines that are responsible for maintaining aircraft;
- Diagnostic system: the airlines' maintenance crews, customers of Cfm, will in future make direct use of the diagnostic tool on a portable computer. The system is included on a CD-ROM, that also contains the parts lists that enable some parts to be rapidly identified, as well as the technical documentation.

The system is currently being extended to cover a wider range of aircraft engines, in particular the CFM-56-5C of the Airbus A340 and the CFM-56-5A of the Airbus A320 which contain on board intelligence and generate new maintenance procedures (*e.g.* is the system that is reported as being faulty really faulty, or is the sensor device faulty?).

Another application in maintenance developed using KATE was a help desk for troubleshooting plastic injection press robots by SEPRO Robotique. The benefits were: the creation of a methodology for troubleshooting the robots; a reduction in the number of calls to the help desk; reduced costs of support; reduced number of incorrectly skipped spare parts (SEPRO manufactures about two thousand five hundred robots and exports sixty-five percent of these). Additional systems developed with KATE cover the maintenance of equipment for manufacturing semi-conductors, maintenance of military vehicles and of power stations.

11.3.2 Application II - Assessing Wind Risk Factors for Irish Forests

This application was developed by IMS for its client Co•llte. The main reason for presenting this application is that it was entirely developed by a third party with only slight involvement from AcknoSoft.

Forestry in Ireland is a rapid growth area of land use alternative to agriculture. After a relatively long period of socially-motivated forestry on marginal land, more commercial management practices have

been established, under the semi-State agency Co•llte. A comprehensive relational database system has been set up, containing details of the life-histories of some 180,000 distinct plots of land. There remains however the problem of how to gain access to corporate experience of best practice, in a situation where the environment is quite different to that of continental Europe, due to the wind regime and mild winters. The wind question has been the subject of some dedicated research into factors governing wind damage; these would appear to include methods of thinning and ground preparation. The following tasks are addressed in support of field decisions:

1. For a new plot, carry out a similarity search over a case-base and select an appropriate species and planting regime;
2. For a current plot after establishment, specify the planting, do a similarity search over plot and planting variables and select from experience appropriate management practice;
3. At each successive management episode (basically, thinning), assess what the current best practice is, with particular reference to exposure to wind damage.;
4. It is projected, as a further development, to interface the case base with a bio-techno-economic model, enabling detailed quantitative adaptation of historic case material to the current case.

Other applications of KATE in biological domains include pollen classification (Elf Aquitaine production), tomato diseases diagnosis (Institut National de Recherche Agronomique, INRA), sponges and nematodes identification (Museum of Natural History in Paris), analysis of Gene Sequences (Genethon), advice on treatment for poisons and psychotropes (Russian advisory board for Toxicology).

11.4 REMIND

11.4.1 Application I - Help Desk for overcoming hardware and software problems in the UK's department of social security

The Information Technology Services Agency (ITSA) in the UK is responsible for providing technical computer support to government offices around the UK. Currently they must support users of thirteen thousand terminals and PC's in overcoming hardware and software problems they encounter on a daily basis. Their goal was to build an automated help desk that could be used by the sixty people who man their customer service hot-line, receiving about one thousand calls per day. They support approximately seventy-five thousand users in the agency and the questions and problems cover about thirty-seven separate "domains" ranging from Lotus 1-2-3, to printers and peripherals, to "how to properly fill out on-line benefit eligibility applications". They were enthusiastic about the idea of being able to recall past cases as a means of solving current problems since there was a very high recurrence of similar problems over time.

Initial prototypes involved cases that consisted of both short freeform problem descriptions and some discrete fields of information like error codes, terminal type, OS, network, *etc.* The system needed to demonstrate an ability to retrieve cases accurately and quickly and to use the freeform problem description information for indexing and retrieval purposes. REMIND's hybrid inductive/nearest neighbour matching approach was able to consistently retrieve relevant cases from the case base. The

production application was built using Visual Basic as the front-end GUI tool talking to two REMIND case libraries via the REMIND API for Windows. Cases handled during the course of the day are stored in an Ingress relational database for call tracking purposes. The two case libraries are divided along hardware and software problem categories. Each case base has upwards of one thousand cases and the system provides an average response time of eight to ten seconds, which is sufficient for handling customers over the phone.

Over the sixteen months that the system has been in operation, ITSA has found that it is much better equipped to handle hardware problems than software problems. This is because hardware problems tend to be more narrowly scoped than software problems. This makes it easier to collect a representative set of cases that cover the majority of problems that users may encounter. In the software area, the case base would need to be much larger and probably require a much more complex representation to get performance comparable to the hardware case base. At present, about sixty percent of all calls are concerned with hardware.

11.4.2 Application II - Process control for Naheola Mill

James River Corp.'s Naheola Mill in Pennington, Alabama is one of the relatively few fully integrated pulp-to-paper production plants in existence. The Mill produces three basic outputs: tissue, board and processed pulp. Tissue is used for Brawny Paper Towels, toilet tissue and other consumer products; board is used in Dixie Paper cups, plates and other consumer products; processed pulp is redistributed to other mills to produce their end-products. At the front end of this huge processing facility, hardwood and softwood trees are pulverised into chips, then fed into the digesting process. Each batch process can consume millions of dollars of input and output is traditionally fairly erratic. The process only produces the targeted output 60% of the time. Wastage can be high and missed outputs are a regular occurrence making inventory management a problem. A great deal of effort to automate the monitoring process and achieve high levels of computerised decision support is expended in order to optimise the production. The Nahoma Mill has worked with several techniques, such as neural networks, rule based systems and fuzzy logic, but they concluded that much of the process depended to a great extent on the judgement and experience of the Mill operations staff to detect batches going awry. They turned to CBR with REMIND. REMIND has been used on one of the many processing steps that involved the operation of a Pulp Dryer. The objective is to maximise the continuous production from the pulp dryer and minimise the rate of change on the Kamyr Digester in the process. Monitored data inputs/states for controlling this relationship number nearly two hundred. Thousands of cases were compiled on previous runs where the outputs were judged for quality and composition. REMIND temporal field representation needed to be extended to handle data monitoring. The outcome is a decision on whether to shut the process off or not. In effect, REMIND provides a "fuzzy" alerting system that alerts operators if something appears to be going wrong. Assuming prompt action on the part of operators, process sub-optimisation can be contained or rectified, resulting in substantial savings throughout the process.

11.5 S^3-CASE

The S^3-CASE version that was provided was a pre-release product. For that reason there are no industrial applications yet.

Chapter 12: Future Trends for CBR

The development trend of CBR methods can be grouped into four main topics: integration with other learning methods, integration with other reasoning components, parallelism and relations to cognition. The current trend of CBR applications is in the help desk domain. The strong role of user interaction, of flexible user control and the drive towards interactiveness of systems favours a case-based approach to intelligent computer-aided assistance - learning, training and teaching.

For complex application domains, simple approaches to domain modelling are not sufficient. Such domains require different problem solving strategies. The automation of the knowledge acquisition process and the ability to automatically adapt to a given environment generates a need for automated learning. Thus, one central task here is the combination of the required problem solving and learning strategies. CBR offers a first (generic) suggestion for the integration of problem solving and learning, for which a deeper analysis from a practical perspective can achieve interesting results. The INRECA European project (Esprit III contract P6322; Althoff, Wess *et al.*, 1994) has already taken steps in this direction. INRECA's basic technologies are induction and CBR. It offers tools and methodologies for developing, validating and maintaining decision support systems. INRECA fully integrates both induction and CBR into a single environment and uses the respective merits of both technologies.

For this purpose, four possible levels of integration between the inductive and the CBR technologies have been identified. The first consists simply of keeping both technologies separate and letting the user choose select one. This toolbox approach should not be rejected because a user may need only one of the two. In the second level of integration, called the co-operative approach, the technologies are kept separated but they collaborate: each uses the results of the other to improve or speed up its own results, or both methods are used simultaneously to reinforce the results. For instance, CBR can be used at the end of the decision tree when some uncertainty occurs. In INRECA, communication of results between the component systems is achieved through the CASUEL language. The third level of integration, called the workbench approach, goes a step further: the technologies are separated but a "pipeline" communication is used to exchange the results of individual modules of each technology. For instance, the CBR system produces a similarity measure between a set of cases that may be used by the inductive system to supplant the information gain measure. The final level of INRECA aims at reusing the best components of each method to build a powerful integrated tool, which avoids the weaknesses of each separate technology and preserves their advantages. Figure 12.1 summarises the four integration levels.

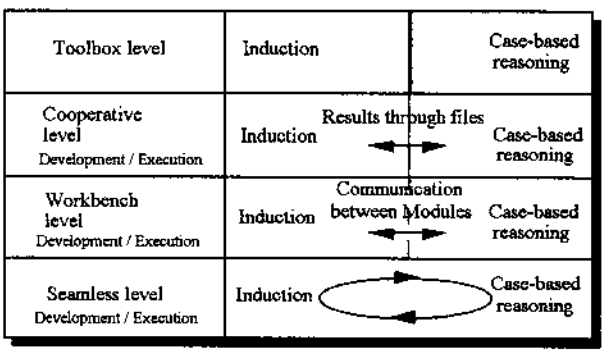

Figure 12.1. Four Integration Levels in INRECA

Beyond the application domains that can be tackled successfully by means of a case-based indexing and retrieval approach and that can be built using today's CBR tools, there are also some other application domains that cannot be tackled without a significant investment in knowledge-based adaptation mechanisms. These other application domains can be characterised as domains where it is extremely unlikely that a former solution is reusable without any substantial modification. These characteristics hold for all non-trivial planning and design problems. Moreover, in much of these planning and design domains, the initial problem description is heavily under-specified and this deficit can only be compensated for by interaction with the user.

Examples of research projects where such issues are tackled are PRODIGY (Veloso, 1994) in the planning area or FABEL (Bakhtari. & Bartsch-Spörl, 1994) in the design area. An important characteristic of FABEL is that it does not aim at solving an architectural design problem completely in one step, but that it offers dozens of little useful design aids, which are embedded into the architect's natural work flow in such a way that they can be activated "in flight". For example, if a user of the FABEL system is not sure about how to proceed, he can just mark the problem zone and the system offers solutions to similar situations on a single mouse click. Then the user can decide which of the suggested solutions to take up and how to adapt it. FABEL is also expanding the scope of the CBR approach to the retrieval of cases that are represented as two or three dimensional sketches or CAD plans. For the most part, these describe complex artefacts geometrically and can be automatically indexed and compared according to their sets of recognisable parts and their structural similarity.

This makes CBR approaches usable for mechanical and architectural engineering tasks and embeds CBR support functions into larger design support environments. In the end it enables the reuse on demand of the know how encoded in former construction layouts and CAD plans. This not only enhances the productivity of the designers, but also helps them avoid known mistakes and shortens the time to market for new products.

What seems necessary is to have CBR systems that are focused on specific tasks (for example help desks). These are able to handle a wide variety of applications, but they directly support particular needs and are not as generic as general purpose shells. The integration issue also appears to be promising. The more complex the real world applications, the greater the need for having closely integrated systems. We believe that in the future there will be less of a need for generic standalone tools, but that we will see a lot more integration of CBR technologies into broader systems. As an example of this, Inference recently announced its intention to integrate CBR EXPRESS (using DDE's) with a software product that does call tracking. Also, the Carnegie Group has announced its intention to integrate CBR capabilities into its Test Bench diagnosis tools.

From a research perspective, the CBR community throughout the world seems very active and enthusiastic about its work. This is clear from the number or CBR events being organised over the next year or so (see under **Events** in Appendix 4: Other Useful Contacts)

ACKNOWLEDGEMENTS

The authors are indebted to Harald Holz, Alexandre Meissonnier, Carsten Priebisch and Wolfgang Wilke for carrying out the tests at the University of Kaiserlautern. Their devoted work was the basis for the success of the evaluation presented here. Thanks also to Stefan Wess for many helpful comments during the evaluation process. We wish to thank all the reviewers of this Report who made invaluable comments draft versions. Thanks to Gerhard Strüne (University of Freiburg, Germany) and Brice Lepape (CEC) who provided helpful comments on early draft; particular thanks to Brigitte Bartsch-Spörl (BSR Consulting, Germany) and Agnar Aamodt (University of Trondheim, Norway) for the quality and detail of the work they did in reviewing the pre-publication draft; and thanks also to Robin Muire (Rolls Royce, England), Rick Magaldi (British Airways, England) and Ian Watson (University of Salford, England) who all provided comments on the pre-publication draft. Finally, we would like to thank Alex Goodall of **AI** *Intelligence* who was perfect in his role of an unreasonably demanding editor and who reminded us constantly of the constraints involved in publishing .... We will hate him for a very long time (at least one week) and, hopefully, everything will be forgiven.

Funding for this work was provided by the Commission of the European Communities (ESPRIT contract P6322, the INRECA project - see Appendix 4).

And of course, we are grateful to AcknoSoft S.A. (KATE), Inference GmbH/Inference, Inc. (CBR EXPRESS), Cognitive Systems, Inc. and its distributor Scientific Computers (REMIND), Esteem, Inc. (ESTEEM) and tecInno GmbH (S^3-CASE) for providing us with access to their tools.

APPENDIX 1: DESCRIPTION OF TESTS

TEST T0 Ability to handle structured domains

- MARINE SPONGES domain
  - capturing the requirements of the domain during a one-day meeting with an expert of the domain
  - analysing all the CBR tools to determine whether they can be used to model the MARINE SPONGES domain such that all the requirements of the expert are met
  - *result*: the MARINE SPONGES domain can be represented (yes/no)

TEST T1 Systematic introduction of noise during consultation

- CAR Domain
  - take all 205 cases as reference cases and choose 50 cases at random as new cases
  - introduce noise for the available attribute values of the new cases (*i.e.* randomly selecting a possible value from the respective value range for n % of the attributes) with the following degrees: 10%, 20%, 40% and 60%
  - compute the percentage of correct classifications for each "degree of noise" (*e.g.* 35 new cases with 10% noise are correctly classified and 15 new cases not: this results in a value of 70 %)
  - *result*: percentage of correctly classified cases for each of the 4 degrees of noise

TEST T2 Systematic reduction of available information within the new case during consultation

- CAR Domain
  - take all 205 cases as reference cases and choose 50 cases at random as new cases
  - delete the available values of n % of randomly selected attributes of the new cases with the following degrees: 10%, 20%, 40% and 60%
  - compute the percentage of correct classifications for each "degree of reduced information" (*e.g.* 45 new cases with 10% reduced information are correctly classified and 5 new cases not: this results in a value of 90%)
  - *results*: percentage of correctly classified cases for each of the 4 degrees of information reduction

TEST T3 Retrieval time[1]

- TRAVEL AGENCY domain
  - randomly select 50 cases as new cases from the 1,470 original cases; use the remaining 1,420 as reference cases
  - build 4 different case libraries of 200, 400, 800 cases (randomly selected from the remaining 1,420 cases, respectively) and the whole case base of 1,420 cases
  - FOR N = 200, 400, 800, 1,420 DO:
    - Retrieve the most similar case from the case base of N cases for the 50 new cases, respectively and measure the retrieval time

---

[1] TESTS T3, T7, and T10 are based on the same distribution of reference and new cases.

- Retrieve the five most similar cases from the case base of N cases for the 50 new cases, respectively and measure the retrieval time
- Retrieve the ten most similar cases from the case base of N cases for the 50 new cases, respectively and measure the retrieval time

- *results*:

  For each of the 4 different case libraries of 200, 400, 800, 1,420 cases:
  - Time needed for retrieving the most similar case (average of 50 tests)
  - Time needed for retrieving the five most similar cases (average of 50 tests)
  - Time needed for retrieving the ten most similar cases (average of 50 tests)

TEST T4 Classification of identical cases

- TRAVEL AGENCY domain
  - randomly select 50 cases as new cases from 1,470 reference cases
  - retrieve the most similar case from the case base of 1,470 reference cases for the 50 new cases, respectively and compute the percentage of correctly classified cases
  - *result*: percentage of correctly classified cases (average of 50 tests)

TEST T5 Compulsory exercise of similarity measures

- CNC MACHINE TOOL domain
  - all CBR tools use the same simple similarity measure (no weighting, no additional knowledge, no combination of different strategies)
  - two reference cases:

    CaseCorrect:
    - ErrorCode = i59
    - I/OStateOut7 = logical-1
    - Valve5Y1 = switched
    - I/OStateOut24 = logical-0
    - Valve5Y2 = not-switched
    - PipesClampingReleaseDevice = OK
    - I/OStateIn32 = logical-1
    - DIAGNOSIS = IOCardIN32i59Defect

    CaseIncorrect:
    - ErrorCode = i59
    - Valve5Y1 = not-switched
    - I/OStateIn32 = logical-1
    - DIAGNOSIS = MagneticSwitch5Y1Defect

  - assumption:
    - the right diagnosis is **IOCardIN32i59Defect**
  - the new case:
    - ErrorCode = i59
    - Valve5Y2 = not-switched
    - I/OStateIn32 = logical-1
  - *subtest 1*: retrieve the most similar case

- take all 311 cases of the CNC MACHINE TOOL domain as reference cases plus CaseCorrect and CaseIncorrect, *i.e.* altogether 313 reference cases
- assumption:
  - the right diagnosis is **IOCardIN32i59Defect**,
  - there are two cases that include IOCardIN32i59Defect as diagnosis (CaseCorrect, CASE 138)
- *subtest 2*: retrieve the two most similar cases

Results from both subtests:

- retrieved most similar case (subtest 1)
- retrieved two most similar cases (subtest 2)

TEST T6 Voluntary Exercise of Similarity Measures

- CNC MACHINE TOOL domain
  - all CBR tools can use the following improvements:
    - causal rule:
      IF ErrorCode = i59 & Valve5Y2 = not-switched
      THEN I/OStateOut24 = logical-0
    - applying test selection to increase the available information:
      - questions for attributes of CaseCorrect are answered as included in this case
      - all other questions are answered with *unknown*
    - any other approach to achieve a correct result, *e.g.* special handling of unknown values
  - take the same new and reference cases as in the "compulsory exercise" (subtest 2)
  - retrieve the two most similar cases with each of the "improvements", respectively
  - *results*: two most similar cases for each kind of "improvement"

TEST T7 Repeating the same queries[1]

- TRAVEL AGENCY domain
  - randomly select 50 cases as new cases from the 1,470 original cases; use the remaining 1,420 as reference cases
  - retrieve the most similar case for the 50 new cases; perform twice and compute the percentage of identical results in both retrieval steps
  - *result*: percentage of identical results (average of 50 tests)

TEST T8 Systematic introduction of noise during application development

- CAR Domain
  - take all 205 cases as reference cases and randomly choose 50 as new cases
  - introduce noise for all 205 reference cases (but not for the new cases) by adding noise to the available attribute values of the reference cases (*i.e.* randomly select a possible value from the respective value range for n % of the attributes) with the following degrees: 10%, 20%, 40% and 60%
  - build an execution system from the reference cases for each degree of added noise

---

[1] Tests T3, T7, and T10 are based on the same distribution of reference and new cases.

- compute the percentage of correct classifications for each "degree of noise" (*e.g.* 40 new cases are correctly classified by an execution system built from noisy reference cases (*e.g.* 10%) and 10 new cases not: this results in a percentage of 80 %)
- *result*: percentage of correctly classified cases for each of the 4 degrees of noise

TEST T9 Systematic reduction of available information within the reference cases during application development

- CAR Domain
  - take all 205 cases as reference cases and randomly choose 50 as new cases
  - delete the available values of n % of randomly-selected attributes for all the 205 reference cases (but not for the new cases) with the following degrees: 10%, 20%, 40% and 60%
  - build an execution system from the reference cases for every "degree of reduced information"
  - compute the percentage of correct classifications for each "degree of reduced information" (*e.g.* 25 new cases are correctly classified by an execution system built from incomplete reference cases (*e.g.* 10%) and 25 new cases not: This results in a percentage of 50 %)
  - *result*: percentage of correctly classified cases for each of the 4 degrees of reduced information

TEST T10 Building speed for automatic application development[1]

- TRAVEL AGENCY domain
  - build the execution system from a set of 200, 400, 800 and 1,470 reference cases and measure the time needed
  - compute the average time per case
  - *results*: building speed of the execution system for the 4 different sets of reference cases, average time needed per case for each of the 4 different sets of reference cases

[1] Tests T3, T7, and T10 are based on the same case distribution of reference and new cases.

APPENDIX 2: LOCAL SIMILARITY MEASURES

In this table, a and b stand for the set of values taken by two cases. The valuation indicates whether an attribute has to take a single value at a time or whether it can takes a list as its value.

	Local Similarity	Attribute Type	Valuation
i.	$\mathrm{Sim}(a, b) = \begin{cases} 0, \text{if } a \cap b = \varnothing \\ 1, \text{if } a \cap b = \varnothing \end{cases}$	symbolic	monovalued multivalued
ii.	$\mathrm{Sim}(a, b) = \dfrac{\mathrm{Card}(a \cup b) - \mathrm{Card}(a \cap b)}{\mathrm{Card}(a \cup b)}$	symbolic	multivalued
iii.	$\mathrm{Sim}(a, b) = \dfrac{\mathrm{Card}(a \cup b) - \mathrm{Card}(a \cap b)}{\min(a \cup b)}$	symbolic	multivalued
iv.	$\mathrm{Sim}(a, b) = \dfrac{\mathrm{Card}(a \cup b) - \mathrm{Card}(a \cap b)}{\max(a \cup b)}$	symbolic	multivalued
v.	$\mathrm{Sim}(a, b) = \dfrac{\mathrm{Card}(a \cup b) - \mathrm{Card}(a \cap b)}{\mathrm{Card}(\mathrm{O})}$	symbolic	multivalued
vi.	$\mathrm{Sim}(a, b) = \dfrac{\mathrm{ec}(\min(a^-, b^-), \max(a^-, b^-)) - \mathrm{Card}(a \cap b)}{\mathrm{Card}(\mathrm{O})}$	ordered symbolic and numeric	multivalued
vii.	$\mathrm{Sim}(a, b) = \dfrac{\lvert a - b \rvert}{\mathrm{ec}(\mathrm{O})}$	numeric	monovalued
viii.	$\mathrm{Sim}(a, b) = \dfrac{\lvert a_c - b_c \rvert}{\mathrm{ec}(\mathrm{O})}$	numeric	multivalued
ix.	$\mathrm{Sim}(a, b) = \dfrac{\mathrm{ec}(\min(a^-, b^-), \max(a^-, b^-)) - \mathrm{ec}(a \cap b)}{\mathrm{ec}(\mathrm{O})}$	numeric	multivalued
x.	$\mathrm{Sim}(a, b) = \dfrac{\mathrm{ec}(a \cup b) - \mathrm{ec}(a \cap b)}{\mathrm{ec}(a \cup b)}$	numeric	multivalued
xi.	$\mathrm{Sim}(a, b) = \dfrac{\mathrm{ec}(a \cup b) - \mathrm{ec}(a \cap b)}{\min(\mathrm{ec}\, a, \mathrm{ec}\, b)}$	numeric	multivalued
xii.	$\mathrm{Sim}(a, b) = \dfrac{\mathrm{ec}(a \cup b) - \mathrm{ec}(a \cap b)}{\max(\mathrm{ec}\, a, \mathrm{ec}\, b)}$	numeric	multivalued
xiii.	$\mathrm{Sim}(a, b) = \dfrac{2 * \mathrm{h}(a \cup b) - (\mathrm{h}(b) - \mathrm{h}(a)}{2 * \mathrm{h}_{\max}}$	symbolic and numeric	multivalued
xiv.	$\mathrm{Sim}(a, b) = \dfrac{\mathrm{h}(\text{node merging } a \text{ and } b)}{\text{total height of tree h}}$	symbolic and numeric	monovalued

- **Monovalued** the attribute has exactly one value at a time;
- **Multivalued** the attribute takes a list as its value or possibly an interval;
- **O** set of possible values for the attribute;
- **Card** size of a set;
- ***a*-, *b*-** upper bound of a (respectively of b);
- $\mathbf{a_c}$ central point of interval a;
- **ec(I)** absolute value between the upper and the lower bounds of interval I;
- **h** level height in a symbol hierarchy.

APPENDIX 3: PRODUCT INFORMATION FROM SUPPLIERS

This section contains information supplied by seven vendors of CBR tools, including those reviewed in this Report. It relates to the versions of the products as at February 1995. The text supplied by the vendors has been reproduced here with only layout and minor editorial changes.

CBR 2 (incorporating CBR Express and CasePoint)

Reviewed in this Report

First Release:
1991

Current Version:
2.0

Platforms for Development:
Windows

Platforms for Deployment:
Windows, OS/2, Sun Solaris, HP-UX

Example Pricing Information:
£15,000 for a 10-user licence

Developed by:
Inference Ltd

Number of employees:
220+

Description of product and major features:

CBR 2 (Incorporating CBR EXPRESS and CASEPOINT)

A complete family of products and services focused on the application of case-based retrieval (CBR EXPRESS) technology, targeted at the help desk and call centre market. CBR 2 will support the entire cycle of customer support - problem management, information retrieval, problem resolution and call avoidance. It enables organisations to capture unstructured corporate data such as expertise, knowledge and multi-media documentation, providing an intuitive interface that enables users to find relevant information very quickly even if they are unfamiliar with the topic area.

Case-Based (Reasoning) Retrieval - as optimised in CBR EXPRESS - lets developers (and end-users) use natural language and a graphical, fill-in-the-form user interface to build knowledge. Since it requires no programming, new cases can be added dynamically, improving the application over time as new data increases the system's depth of information. It is suited to a wide range of applications from engineering and sales to customer support and finance and requires little training.

Update in relation to version used in this Report

Since the product evaluation in this Report was undertaken, we have released a new version of the product that opens up new applications and addresses some of the points raised in the review. We now offer a family of products that includes CBR EXPRESS under the family name of CBR 2. Included within the CBR 2 family are:

- CBR EXPRESS HELP DESK SERIES 2.0: A complete problem management and problem resolution system aimed at Help Desk users
- CASEPOINT 2.0: This is a case base view for end-users to access case bases created by CBR EXPRESS. CBR EXPRESS is now used for authoring rather than deployment. CASEPOINT runs in native mode under Windows, Presentation Manager, Solaris and HP-UX. In addition there is a DLL version for embedding into other applications.
- CBR EXPRESS 2.0: This is essentially the same as the version in the review, but there are some new modules:
  - Generator: for automatically creating case bases from a series of documents - either .txt or MS Word .doc files. It allows CBR 2 to navigate document case bases as well as problem diagnostic case bases.
  - Tester: for testing and auditing case bases, checking their design and highlighting possible areas for investigation.
  - Pre-built case bases: Inference now offers a wide range of pre-built case bases covering most popular PC products within a Windows environment.

ART*ENTERPRISE

For more complex applications, ART*ENTERPRISE provides a richer set of CBR functions.

ART*ENTERPRISE is a single, comprehensive development tool that integrates the functionality of the numerous separate products that are essential for creating strategic corporate wide applications. It provides rapid prototyping, development and deployment: object-oriented programming; built-in graphical user interfaces (GUIs) with multimedia capabilities; data integration and modelling from multiple DBMSs; event-driven, client server, open architecture; access to unstructured information; business rule processing with "what if" analysis; and facilitates business process reengineering and downsizing.

Comments from Inference relating to the report:

General point: CBR EXPRESS is an application shell designed to be used by business users rather than specialist programmers. Therefore, CBR functionality has been kept simple to ensure ease of use and the creation of successful case bases by non-technical staff. ART*ENTERPRISE provides a richer set of CBR functions together with fully obect-oriented and rule based programming. As such it is more comparable with some of the other products in the review.

Applications: Although the review rightly suggests that Help Desks is an important application area, the bulk of our customers are using the product for facing Call Centres.

Rules: CBR EXPRESS does in fact support rules - although in the version used during the review this was an undocumented feature! The user can build rule files that automatically answer questions. The rules can pick up words from textual input to pre-answer questions. Also rules can be used to infer answers to other questions from previously answered questions. For example, if the answer to "Is the light on?" is "Yes", a rule can automatically answer the questions "Is the power on?" with "Yes".

Performance: Although CBR EXPRESS performed well, CASEPOINT, which is used for deploying case bases, performs significantly faster, utilising less memory.

Customising the user interface. CBR EXPRESS is now positioned as the authoring environment, whilst CASEPOINT is used for deploying applications. CASEPOINT has a complete set of DDE functions that enable it to interface with most GUI building products such as Visual Basic. Also there is a DLL version for completely embedding in third party products

Distributed by:

UK: **Inference Ltd**
31-37 Windsor Road
SLOUGH
Berkshire, SL1 2EL
United Kingdom
Tel: +44 1753-811 855 Fax: +44 1753-811 860

Germany: **Inference GmbH**
Lise-Meitner Straße 3
85716 UNTERSCHLEISSHEIM
Germany
Tel: +49 89-321 8180 Fax: +49 89-321 81830

North America: **Inference Corporation**
101 Rowland Way, Suite 310
NOVATO
CA 94945
USA
Tel: +1 415-899 0200 Fax: +1 415-899 9080

Inference has also appointed distributors and partners for most other countries. Initial enquiries should be addressed to the appropriate Inference office who will direct enquiries to the appropriate partner. For enquiries outside North America please contact: Inference Ltd. For North America enquiries, please contact Inference Corporation.

ESTEEM

Reviewed in this Report

First Released:
1991

Current Version:
1.4

Platforms:
Windows 3.1

Example Pricing:
$495 International Sales, academic discounts are available and multiple copy agreements are aggressively discounted.

Developed By:
Esteem Software, Inc.

Number of employees:
10

Description of product and major features:

General description

ESTEEM 1.4 is a Windows-based software tool that enables individuals (both programmers and npn- programmers) to quickly construct decision enabling applications which utilise CBR technology. ESTEEM delivers these capabilities in a form which makes it easy to learn and apply. Five simple editors are provided:

1) to help the developer define their case-based system,

2) to create the definition of how to assess similarity for retrieving past cases,

3) a rule-base editor for creating rules used in retrieval and adaptation,

4) a case editor for entering new cases or for using existing databases and ASCII files for cases,

5) an end-user interface editor for creating simple interfaces to the case-based application.

Many methods for fine tuning the retrieval process are provided ranging from multiple forms of string matching, fuzzy numeric matching, ID_3 and Gradient Descent weighting, to nested case-base retrieval.

Developers can embed ESTEEM in other Windows applications using its DDE library and database connections.

Update in relation to version used in this Report

ESTEEM 1.4 has a new Automatic Weight Generation facility. This facility provides two different heuristics for helping the user to determine the relative weights of the case base's features when the "Weighted Feature Similarity" similarity type is used. The user can specify one of two weight generation methods: the ID_3 method or the Gradient Descent method.

ID3 Weight Generation Method

The ID3 method currently works only for features using the Exact match type (or, in the case of numeric features, the Equal match type). The algorithm builds a decision tree for the cases in the current case base by using the ID_3 algorithm and then uses the tree generated to calculate weights for the features that were used in the tree.

After selecting the ID_3 method, the user specifies one target feature (the feature the generated tree could be used to "predict") and the source features to use (*i.e.* the features that will be used in the generated tree to predict the target feature). ESTEEM will then generate a decision tree.

Gradient Descent Weight Generation Method

Unlike the ID_3 weight generation method, the Gradient Descent method works for all feature and match types. As in the ID_3 method, the user specifies target and source features. However, more than one target feature may be specified for this method.

The method's algorithm works as follows: several random cases are selected from the case base and the cases that are most similar to them (based on the current weights of the source features) are found.

Information on how much the weights of the source features should be incremented or decremented is calculated, based on how well the matching cases' source feature values match, as well as how well the matching cases' target features values match. After examining several random cases, the resulting "weight updates" vector is normalised, scaled by a factor Delta and added to the current source weight vector. The factor Delta is then decreased and the algorithm begins examining more

random cases. This process continues until Delta reaches a certain value or until the user tells ESTEEM to stop.

New Multimedia Feature Type Supported

ESTEEM 1.4 supports a new Multimedia feature type. ESTEEM can be made to execute the appropriate program with the displayed file. Two match types are supported for Multimedia features: Exact (case indifferent) and Inferred.

New Match Types Supported

In addition to the match types available in ESTEEM 1.3's similarity definition, ESTEEM 1.4 now supports two new match types for numeric features: Absolute Range and Absolute Fuzzy Range.

These two match types are similar to the Range and Fuzzy Range match types, except that the similarity between two values are now measured in terms of the absolute difference between the two values rather than the percentage difference.

Two new match types, Subset and Superset, are also available to facilitate the use of text features as multi-valued sets. The Subset matching function returns the percentage of the target case's value's elements - as delimited by the spaces and commas in the text - that appear in the current case's values. Likewise, the Superset matching function returns the percentage of the current case's value's elements that appear in the target case's value.

An additional function available within rules, GetNthElement, provides access to the text features' individual elements, as delimited by the spaces and commas in the text.

List of major distributors:

International: **Esteem Software Incorporated**
302 E. Main St.
CAMBRIDGE CITY, IN 47327
USA
Tel: +1 317-478-3955 Fax: +1 317-478-3550 Email: esteem13@delphi.com
Contact: Jill S. King

Far East: **Axon, Inc.**
9F, No 259,
SEC.2 Ho-Ping East Road
TAIPEI 106
Taiwan
Tel: +886 2-704 5535 Fax: +886 2-754 1785

Belgium: **Impakt nv**
Belgium
Fax: +32-91 33 00 78
Contact: Frank Lateur

CASECRAFT, THE KATE TOOLS (KATE-INDUCTION, KATE-CBR, KATE-EDITOR, KATE-RUNTIME)

Reviewed in this Report

First released:
1988

Current version:
4.0

Platforms for development:

PC Windows, 8Mb of memory

Platform for deployment:

PC Windows, Macintosh, SUN

Example Pricing Information:

Call

Developed by:

Michel Manago

Number of Employees:

Not specified

Description of product and major features:

- KATE-INDUCTION allows object representation of cases and imports most database and spreadsheet formats. It includes a dynamic induction module that generates tests "on the fly" for better treatment of unknown values at consultation time. It offers interactive data mining tools and includes utilities to print the tree on a variety of paper formats. KATE combines inductive capabilities with object-oriented case representation and numerous extensions such as the use of background knowledge.
- KATE-CBR performs nearest neighbour matching and does full fuzzy matching (unlike the version of KATE 3.0 that was evaluated in this report). KATE-CBR supports the same object language as KATE-INDUCTION and offers customisation facilities for the similarity measure. KATE-CBR can be combined with KATE-INDUCTION.
- KATE-EDITOR allows the developer to build an object model of the cases and uses that model to generate an interactive questionnaire to edit the case library. KATE-EDITOR is a set of C DLL's that run on top of Asymetrix' TOOLBOOK (included with KATE) to edit user interfaces and for multimedia extensions.
- KATE-RUNTIME is used to deliver consultation stacks that can be edited with TOOLBOOK

KATE 4.0 strong points are:

1. Speed and ability to handle large case bases that have been drastically improved over previous versions of KATE such as the one reviewed in this report. Performance measures (on a DX2/66 PC) show that KATE-CBR achieves pure nearest neighbour retrieval in less than 2 seconds for a case databases with over 8,000 cases described by 50 attributes without pre-indexing. KATE-INDUCTION generates decision trees in about 10 seconds for the same case base.
2. The ability to integrate with other programs, an open architecture, client server capabilities and ease of modification of the user interface.
3. Combinations of technologies to configure and tune the architecture of the CBR system to suit the application requirements: inductive retrieval, nearest neighbour, dynamic indexing. The system either retrieves cases using full description or queries the user for answers in a step by step mode. Automatic test facilities are included to evaluate the system's accuracy before it is fielded.
4. Low-cost runtime that support all the features of the development environment (except case structure modification and import facilities).
5. Use of background knowledge, ability to represent complex cases (in particular for engineering tasks) and integration of numeric and symbolic learning techniques.

List of major distributors:

USA: **AcknoSoft**
396 Shasta Drive
PALO ALTO
CA 94306-4541
USA
Tel: +1 415-856 8928 Fax: +1 415-858 1873

International: **AcknoSoft**
58 rue du Dessous des Berges
75013 PARIS
France
Tel: +33 1-4424 8800 Fax: +33 1-4424 8866

Belgium (banking applications only)
Call for name of distributor

Switzerland
Call for name of distributor

Ireland
Call for name of distributor

Russia (banking applications only
Call for name of distributor

USA(process and quality control applications only)
Call for name of distributor

Germany
Call for name of distributor

MEM-1

First Released:
1993

Current Version:
1.0

Platforms for Development:
(intentionally left blank)

Platforms for Deployment:
1) IBM 386/486, running Windows
(requires GCLISP 4.3, CLISP, or Franz Allegro LISP [16-bit])
2) DECStations running ULTRIX (requires Lucid Common LISP 4.0)
3) IBM RS/6000 running AIX (requires Allegro Common LISP 4.1.tbeta.0)
4) Macintoshes (requires Macintosh Common LISP 2.0)
5) Suns (requires SUN Common LISP 4.0.1)

Example Pricing Information:
$199.00 single license (discounts for multiple licences)
$50.00 academic purchasers

Developed by:

CECASE (Centre for Excellence in Computer Aided Systems Engineering, at the University of Kansas). CECASE is one of five university-based "centres of excellence" established in the State of Kansas and partially funded by the Kansas Technology Enterprise Corporation with the goal of promoting economic activity in the State. We do research and development and our expertise is in computer-aided design and analysis tools.

Number of Employees:

8 full-time + several part-time researchers and student assistants

Description of Product and Major Features:

MEM-1 is a LISP-based language that allows researchers and developers to build CBR applications. MEM-1 defines case structures and supports indexing, matching, similarity assessment, retrieval and adaptation.

List of Major Distributors in all countries:

All countries: CECASE
2291 Irving Hill Road
LAWRENCE
Kansas 66045-2969
USA.
Tel: +1 913-864 4896 Fax: +1 913-864 7789
Contact: Tony Wei, Software Engineer, Tel: +1 913-864 7743

RECALL

First released:

August 1993

Current version:

1.2

Platforms for development:

SUN, IBM RS6000, HP 9000 series 700, DEC Alpha (under Motif) and PC under Windows.

Platforms for deployment:

As above

Example pricing information:

Development licence: 50,000 FF (£ 6000) per machine (independent of the platform)
Client version: 5000 FF (£ 600)

Developed by:

ISoft S.A.

Number of employees:

14

Description of product and major features:

RECALL is a CBR tool for solving problems or analysing situations by analogy with previous experience. This technique is rapid to initialise and allows the user to build efficient decision help systems that can be easily maintained and enlarged.

Knowledge representation

ReCall has an object-oriented knowledge representation language that allows it to describe and treat cases in "attribute-value" form, as well as in a structured, hierarchical, non homogeneous and noisy form, using knowledge about the application domain. The user builds this structure with specialised graphic editors. Each attribute can be enriched with facets such as "Default", "Cost", "Taxonomy", *etc.* For instance, method facets are programs for computing the value of a descriptor from the values of others. In addition to the hierarchical structure of objects, the user can define relations between several objects.

Creation of cases

When an expert has completed this modelization, he uses the structure as a model for entering new cases. Thus, a case is represented by the set of instances of objects that describe it by the list of relations between these objects.

During the creation of a case, knowledge about the application domain is used to lead and to facilitate input. For example, the "Default" facet and the methods allow the values of some attributes to be set automatically.

Indexing

Indexing is performed automatically by ReCall under the control of the user. One way to control indexing consists of the introduction of knowledge that will be used during the indexing process. This could be attributes whose values are costly to obtain or to define lists of attributes that will not be examined during the process. Finally, a tree editor displays indices and allows their statistical evidence to be controlled and modified by the user.

Similarity

ReCall provides similarity measures taking into account global matching similarity and structural matching similarity.

In the global matching similarity, ReCall computes similarity attribute by attribute, taking into account characteristics of descriptors, knowledge about the domain and modularity links between objects. The global similarity is an aggregation of the similarities for each attribute.

The structural matching similarity takes into account more or less (depending on a parameter defined by the user) the hierarchical structure of cases and the similarity of attributes. The influence of exceptional values can be reinforced when they are considered important, or minimised when they are considered to be noise.

Adaptation

Adaptation is carried out from solutions of cases similar to the current problem. It consists of transforming the solution of these cases to satisfy the constraints of the new problem and proposing solutions to the user. ReCall has standard adaptation mechanisms such as the vote. In addition, the user can define a set of specific adaptation rules. They are application domain dependant and can be considered as knowledge that is stored in the base.

Development of an application

ReCall Tool is a package that uses a graphical user interface in which the methodology of application development is defined.

ReCall C++ Lib consists of using a library which groups CBR functions of ReCall for applications that need particular requirements.

List of major distributors in all countries:

All countries: **ISoft SA**
Chemin de Moulon
91190 GIF SUR YVETTE
France.
Tel: +33 1-6941 2777 Fax: +33 1-6941 2532 Email: recall@isoft.fr
Contact person: Thierry Brunet, Jean Jacques Cannat, Hugues Marty.

France: **Bull SA**

REMIND

Reviewed in this Report

First Released:
July 1992 V1.1

Current Version:
January 1995 V1.3

Platforms for Development:
DOS/Windows; OS2/PM; Macintosh; SUN UNIX

Platforms for Deployment:
DOS/Windows; OS2/PM; Macintosh; SUN UNIX

Pricing Information:
Development System: $3000; API: $2000
Run-Times: $200

Number of Employees:
12

Description of Product and major features:

ReMind

ReMind provides true generic shell capability. It consolidates over ten years of CBR research into a robust, generic capability, having been tested in a diversity of domains to demonstrate its effectiveness in solving a variety of problem-solving tasks. Tools are provided for representing case fields, symbol hierarchies, high level domain concepts and causal relationships. ReMind provides a very rich representational environment. It supports multiple inheritance, partial ordering, qualitative model and the creation of 14 different field types, including symbol, text, integer, float, formula, case, Boolean and date, as well as creating lists of the above field types. Choice of indexing strategies including induction, nearest neighbour and case template. These techniques may be used independently or in combination, providing great flexibility in automating decision support tasks. Case libraries can support several hundred fields and tens of thousands of cases. Cases stored for future reference and retrieval are a gold mine of information. A single library may be used for more than one application for maximum leverage of data. ReMind also includes a database import tool, allowing a database to be transformed into a case base and thus taking advantage of the knowledge in existing data. Compact binary tree are used for case memory. Average retrieval of cases is performed in O(log n) time (where n is the number of cases in the library), providing efficient runtime processing speeds with low memory requirements. Sophisticated tools are provided for analysing, comparing, explaining and generalising about collections of cases.

ReMind's case adaptation capability makes it possible to go beyond a valuable, similar solution by creating formulas that will adapt the retrieved case to suit the new circumstances. It provides built-in tools for testing the accuracy of retrievals and the balance of the case library. These allow validation and fine-tune applications for high performance. ReMind is written in C++ which

provides portability and integration into all environments along with all the benefits of speed and small memory requirements. Its graphical interface fully integrates with Windows, Presentation Manager and Mac OS. It is designed so as to make it as easy as possible for computer and non-computer professionals alike to develop their own custom applications. A form editor allows the rapid and easy design of end-user interfaces with user-friendly English explanation capabilities.

API:

The ReMind Application Programming Interface (API) allows developers of case libraries created in the ReMind Development System to deliver embedded case-based systems to an end user PC, Work Station, or client/server configuration. When the API is deployed in this fashion, end-user access to ReMind's functionality is quick and easy. The API is a fully documented collection of over 50 C functions that allow access to and manipulation of a ReMind case library. In order to access these functions, a Graphical User Interface (GUI) must be built in full with a common GUI tool (*e.g.* Hypercard, Visual Basic or Powerbuilder). The API can also be embedded into applications built with other commercial products (any program or GUI tool) that can link to external C libraries such as automated help desks or expert system applications. This essential developmental flexibility allows developers to create and deploy a custom user interface of a case-based problem solving system to the end-user.

ReMind API Capabilities

- Open, close and create new case libraries, with separate library views.
- Create and edit case library fields.
- Create and store new cases or modify existing cases.
- Create and access symbols in the symbol hierarchy.
- Perform Inductive, Nearest Neighbour and Template Retrievals.
- Perform nested retrievals.
- Create new weight vectors for nearest neighbour matching.
- Create new templates for template matching.
- Perform case adaptation.

Available Function Categories of the API for ReMind

- Managing Libraries and Views Within Libraries.
- Managing Cases.
- Managing Fields Within Libraries.
- Managing and Evaluating Symbols.
- Managing Weight Vectors.
- Managing Templates.
- Performing Case Retrievals.

List of major distributors in all countries:

USA: **Cognitive Systems, Inc.**
220-230 Commercial Street
BOSTON MA 02109
USA
Tel: +1 617-742 7227 Fax: +1 617-742 1139
Contact Person: Dean Burson

Asia Pacific: **Human Interface Engineering Pte. Ltd.**
2 Marsiling Lane
WOODLANDS NEW TOWN
Singapore 2573

Tel: +65 36- 82 242 Fax: +65 36-82 241
Contact: Teck H. Goh

Australia/New Zealand: Gledhill and Associates
89 Church Street, Hawthorn,
VICTORIA 3122
Australia
Tel: +61 38-18 07 95 Fax: +61 38-19 25 99
Contact: Julie Gledhill

Benelux: Case Based Solutions
A.van Schendelstraat 570
3511 MH UTRECHT
The Netherlands
Tel: +31 30 30 49 401 Fax: +31 30 30 49 402
Contact: Aziz Shawky

Brazil: Softon Sistemas de Computadores, Ltda.
Av. Ibitapuera, 2033
CEP 04029-100
SLO PAULO
Brazil
Tel +55 11 549 7833 Fax: +55 11 549 7571
Contact: Reginaldo Martineschen

France: Ingenia SA
92 bis. Av. Victor Cresson
92130 ISSY-LES-MOULINEAUX
France
Tel: +33 1-47 36 29 00 Fax: +33 1-45 29 03 04
Contact: Pierre Vesoul

Italy: AIS SpA
Via Rombon,11
MILAN
Italy
Tel: +39 2-264 0197 Fax: +39 2-264 10744
Contact: Prof. Francesco Gardin

Scandinavia: Computas Expert Systems AS
Leif Tronstads plass 6
PO Box 430
1300 SANDVIKA
Norway
Tel +47 67 54 11 11 Fax: +47 67 54 10 11
Contact: Per Spangebu

United Kingdom/Ireland Intelligent Applications Ltd.
1 Michaelson Square
LIVINGSTON
West Lothian EH54 7DP
Scotland UK
Tel: +44 1506-47 20 47 Fax: +44 1506-47 22 82
Contact: Chris Nelson

S_3-CASE

Reviewed in this Report

First Released: 1994

Current Version: 1

Platforms for Development: Macintosh, Windows, Windows/NT, OS/2, Sun, HP, SGI and other UNIX Platforms

Platforms for Deployment: As above

Example Pricing: Contact vendor

Developed by: tecInno GmbH

Number of employees: 6

Description of product and major features:

S_3-CASE is a tool for the development of CBR applications that actually offers inductive techniques as well as nearest neighbour search capabilities in an integrated way. It supports full object oriented domain modelling as well as multimedia features. The generic user interfaces are adaptable to users needs. The product is supplied as component ware, in the sense that in conjunction with consultancy those parts of the software are selected that are necessary for the customers application. Such a project is usually a 5 stage process starting with a small feasibility test and ending with the exploitation support.

The current version of S_3-CASE includes a forward chaining rule interpreter. This mechanism allows the definition of rules for search restriction, rules for case completion and rules for case adaptation.

List of major distributors in all countries:

World-wide: **tecInno GmbH**
Sauerwiesen 2
67661 KAISERSLAUTERN
Germany
Tel: + 49 6301-60 6 0 Fax: + 49 6301-60 6 66 E-mail: rappi@informatik.uni-kl.de
Contact: Mr Ralph Traphoener

APPENDIX 4: OTHER USEFUL CONTACTS

AI-CBR

An Internet forum maintained by Ian Watson at the University of Salford. The address is: ai-cbr@mailbase.ac.uk

CasePower (formerly known as Induce-It)

CASEPOWER is a PC-based tool sold by Inductive Solutions, a corporation founded in 1989, with four employees. The company, which is based in New York City, builds proprietary models and systems primarily for the financial industry and offers spreadsheet-based development tools.

Inductive Solutions(USA): Tel: +1 212-945 0630; Contact: Roy S Freedman

The Easy Reasoner

The Easy Reasoner from the Haley Enterprise is a tool for software developers that comes bundled as a set of C libraries to add inductive and CBR capabilities.

The Haley Enterprise (USA): Tel: +1 412-741 6420

Events

The AAAI has been holding CBR workshops for some time. Most recently, David Leake organised a workshop at AAAI'93, and David Aha at AAAI'94. In Europe, the first European workshop on CBR (EWCBR) was organised by Prof. Michael Richter near Kaiserslautern, Germany (Wess *et al*, 1994) and the second EWCBR by Mark Keane (University of Dublin, Ireland), Jean Paul Haton (University of Nancy, France) and Michel Manago in Chantilly, France (Keane *et al.*, 1994).

Some of the CBR events during 1995 and 1996 are as follows.

Ralph Barletta, Michel Manago and Stefan Wess (Inference Germany) will organise a workshop on industrial strength CBR applications at IJCAI 95 in Montreal (Canada). The third EWCBR will be held in Switzerland in 1996 and every other year after that. In April 1995, Unicom Seminars will be holding a CBR Workshop, chaired by Alex Goodall. The US and European CBR communities have decided to hold a joint international conference every other year. The first International conference on CBR will be organised in Portugal in October 1995 by Manuela Veloso (Carnegie Mellon University, USA), Agnar Aamodt (University of Trondheim, Norway) and Carlos Bento (University of Coimbra, Portugal).

FABEL (see Chapter 12)

German Research and Development system, funded by the German Ministry for Research and Technology (BMFT, contract n° 01 IW 104) with a particular focus on the architectural domain and on adaptation.

Contact: Brigitte Bartsch-Spörl
BSR Consulting
Wirtstrasse 38, D-81539 München - GERMANY
Tel: +49 89-695545 Fax: +49 89-695158 e-mail: brigitte@bsr-consulting.de

Other partners involved: GMD (Sankt Augustin), Technical University of Dresden, HTWK Leipzig, University of Freiburg, and University of Karlsruhe.

Induce-It

See CasePower.

INRECA (see Chapter 12)

European Research and Development system, funded by the Commission of European Communities (ESPRIT contract P6322), that aims at integrating induction and Case-Based Reasoning with a focus on applications in technical maintenance and classification of natural objects. Lead by AcknoSoft in Paris, the INRECA consortium involves IMS in Ireland, tecInno and the University of Kaiserslautern in Germany.

Contact Michel Manago
AcknoSoft SA
58 rue du Dessous des Berges
75013 Paris - FRANCE
Tel: +33 1-44 24 88 00 Fax: +33 1-44 24 88 66

Other contacts: Michel Baudin, AcknoSoft, USA: Tel: +1 415-856 8928, Fax: +1 415-85 1873; Germany: Ralph Traphoner, tecInno Gmbh Germany: Tel: +49 6301-606 60, Fax: +49 6301-606 66; Sean Breen, Irish Multimedia Systems (IMS), Ireland: Tel: +353 1-284 05 55, Fax: +353 1-284 0829; Klaus-Dieter Althoff, University of Kaiserslautern, Germany: Tel: +49 631-205 3360, Fax: +49 631-205 3357.

INRECA +

Research and Development project between AcknoSoft and the Russian academy of science that focuses on medical domains (INTAS project 4040). The goal is to integrate technologies in multi-media and CBR to develop a help desk for the Russian Information toxicology advisory centre. Based in Moscow, the centre receives calls from all over Russia. It processed more than 6000 calls over the past two years. The similarity between cases is assessed based not only pathology of cases, and on the chemical structures of drugs. A particular emphasis has been made on drug interactions and on psychotrops. This includes children that ingest foreign drugs because they are conditioned in attractive colourful packages, suicide attempts, problems that arise because of drugs that are taken in combination with alcohol *etc.* An editor for chemical structures of compounds and a multimedia database that stores chemical structures and documentation about the psychotrops has been developed to handle real life volume of data about toxic compounds.

Contact: Michel Manago, AcknoSoft, France (see under INRECA above) Oleg Larichev, Institute for System Analysis, Russia: Tel: +95-135 85 03, Fax: +95-938 22 09, Michael Zabezhailo, All-Russian

Institute of Scientific and Technical Information of the Russian Academy of Science (VINITI), Russia Tel: +95-155 43 65, Fax: +95-152 5447, Ioury Zisser, Reliable Software Inc., Belarus Tel: +172 49-40 96, Fax: +172 49-64 83, Y. N. Pechersky, Institute of Mathematics, Moldova, Tel: +373 2-73 81 30.

Recon

Recon is a data mining tool developed at Lockheed to support its servicing contracts with special focus in the financial area. It includes CBR and rule based reasoning technologies, and combines top-down and bottom-up data-mining. Recon can interface with a wide variety of data sources such as Oracle, DB2, Paradox, Rdb *etc.*

USA: contact Evangelos Simoudis
Lockheed AI Centre
3251 Hanover Street
Palo Alto, CA 94304
Tel: +1 415-354-5271, Fax: +1 415-424-3425

Expert Advisor

Expert Advisor, developed by Software Artistry in the US, is a system that does call tracking. Although the supplier claims to have incorporated CBR, Expert Advisor seems to only offers an editor to build decision trees by hand and does not offer tools to built these automatically from cases (induction) or to retrieve them dynamically (CBR). For this reason, we have seen nothing in Expert Advisor that qualifies it as a CBR tool.

Software Artistry: Tel: +1317876 3042

UCI Repository of Machine Learning Databases

Set of public domain databases that are often used to compare performance of Machine Learning and Case-Based Reasoning systems and algorithms. It can be accessed through the Internet, by anonymous ftp at ics.uci.edu, on directory pub/machine-learning-databases/imports-85.

REFERENCES

Aamodt, A. and Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations and System Approaches. *AI Communications* 7(1), pp. 39–59.

Althoff, K.-D., Faupel, B., Kockskämper, S., Traphöner, R. and Wernicke, W. (1989). Knowledge Acquisition in the Domain of CNC Machining Centres: the MOLTKE Approach, in: *Proc. of EKAW 1989*, pp. 180–195.

Althoff, K.-D., Wess, S., Bergmann, R., Maurer, F., Manago, M., Auriol, E., Conruyt, N., Traphöner, R., Bräuer, M. & Dittrich, S. (1994). Induction and Case-Based Reasoning for Classification Tasks. In: H. H. Bock, W. Lenski & M. M. Richter (eds.), *Information Systems and Data Analysis, Prospects-Foundations-Applications, Proc. 17th Annual Conference of the GfKl, University of Kaiserslautern, 1993*, Springer Verlag, Berlin-Heidelberg, pp. 3-16.

Althoff, K.-D. & Wess, S. (1991). Case-Based Knowledge Acquisition, Learning and Problem Solving in Diagnostic Real World Tasks. *Proc. 5th European Knowledge Acquisition for Knowledge-Based Systems Workshop, Glasgow & Crieff, Scotland, UK.*

Auriol, E., Manago, M., Wess, S., Althoff, K.- D. & Dittrich, S. (1994). Integrating Induction and Case-Based Reasoning: Methodological Approach and First Evaluations, in Proc. of EWCBR'94, M. Keane, J.- P. Haton & M. Manago (eds.), AcknoSoft Press, pp. 145-157.

Bakhtari, S. & Bartsch-Spörl, B. (1994). Bridging the gap between AI technology and design requirements. In Gero, J.S. & Sudweeks, F. (eds.): Artificial Intelligence in Design'94. Kluwer, Dordrecht, pp. 753-768.

Bareiss, R. (1989). *Exemplar Based Knowledge Acquisition: A unified Approach to Concept Representation, Classification and Learning*, Academic Press inc.

Bareiss, R. (Ed.) (1991). *Proc. 3th DARPA Case-Based Reasoning Workshop*, San Mateo, California, Morgan Kaufmann Publishers.

Barletta R. and Hennessy D. (1989). Case Adaptation in Autoclave Layout Design, in: *Proc. of the Case-Based Reasoning Workshop*, Pensacola, Florida, Morgan Kaufmann Publishers, pp. 203–207.

Barletta R. (1994). a Hybrid Indexing and Retrieval Strategy for Advisory CBR Systems Built with REMIND, *Proc. of the second European Workshop on Case-Based Reasoning EWCBR'94*, AcknoSoft Press (Paris), pp. 49-58 (To be reprinted and distributed by Springer-Verlag in 1995).

Harmon P. (1992). Overview: Commercial Case-Based Reasoning Products. *Intelligent Software Strategies*, Jan. 1992.

Hennessy D. and Hinkle D. (1992). Applying Case-Based Reasoning to Autoclave Loading, *IEEE Expert Magazine*, Los Alamitos, CA, Oct. 1992, pp. 21–26.

Hennessy D. and Hinkle D. (1991). Initial Results From CLAVIER: A Case-Based Autoclave Loading Assistant, in: *Proc. of the Case-Based Reasoning Workshop*, Washington D.C., Morgan Kaufmann Publishers, pp. 225–232.

Hinkle D. and Toomey C. (1994). *Clavier*: Applying Case-Based Reasoning to Composite Fabrication, To appear in: *Innovative Applications Of AI*, 1994. AAAI Press, San Mateo, California.

Hüber, K., Nakhaeizadeih, G. (1993). Proc. of the Second German Expert Systems Conference (XPS 93), Puppe & Günter (eds.), Springer-Verlag, pp. 167-180.

Keane, M., Haton, J.P., Manago M. eds (1994): Proceedings of the Second European Workshop on Case-Based Reasoning (EWCBR-94), AcknoSoft Press, Paris (revised version will be published by Springer Verlag in 1995).

Kolodner, J. L. (1993). *Case-Based Reasoning*. Morgan Kaufmann.

Perray, M. (1990). Comparative study between three knowledge acquisition techniques to build the same knowledge base: interview, induction and statistical analysis (*in French*: Etude comparative entre trois techniques d'acquisition des connaissances: interview, induction et analyse statistique pour construire une même base de connaissances, *Proc. of the 4th JIIA*, Paris.

Riesbeck, C. K. and Schank, R. C. (1989). *Inside Case-Based Reasoning*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Veloso, M. M. (1994). Planning and Learning by Analogical Reasoning. , Lecture Notes in Articial Intelligence series, Springer -Verlag, Berlin.

Wess, S. Althoff K.- D., Richter M. eds. (1994). Topics in Case-Based Reasoning, Proceedings of the first european Workshop EWCBR-93, Lecture Notes in Artificial Intelligence Series, Springer Verlag, Berlin.

GLOSSARY

Abduction
Method used to infer facts from rules in backward-chaining.

Application Development
Process of using the development system.

Attribute
Feature or component description of a case. An attribute can be of different types: symbolic (ordered or not), taxonomic, numeric (integer and real).

Background Knowledge
Everything important that should be known - facts, relations, procedures, *etc.* - when dealing with a new problem in a specific domain. Usually the domain knowledge is minimal in statistics or neural nets, average in case-based approaches and high in rule-based systems.

Backward-Chaining
Control procedure that attempts to prove facts recursively from inferential rules, first by enumerating conditions that would be sufficient for fact proving and second by setting these conditions themselves as facts.

Case
Specific problem-solving experience. A case usually contains a problem description and a solution. It may be enhanced by additional parts such as a justification, a problem solving path and various types of annotation. Cases can be represented in a computer either as flat attribute-value pairs or as structured objects of different complexity, possibly accompanied by non-symbolic multi-media supplements.

Case Base
Set of cases or examples of a domain, used during problem solving. The case base is usually stored in a database.

Case-Based Reasoning
Method which solves a new problem by recognising its similarities to a specific known problem then by transferring the solution of the known problem to the new one.

CASUEL
Language used for cases and domain knowledge description and exchange, defined during the ESPRIT project INRECA.

Class
See diagnosis.

Classification Task
See diagnosis.

Consultation
Process of using the execution system.

Database

Set of cases stored in a computer memory. The database and the case base are often used as synonyms in CBR literature. However, the database refers to how cases are stored and accessed physically on a storage memory, whereas the case base refers to how cases are used during the CBR process.

Decision Tree

Also called induction tree in this book, a decision tree is a directed acyclic graph which describes a decision process. The inner nodes of the tree represent attributes whose edges correspond to specific values, the final leaves represent the possible decisions.

Deduction

Method used in artificial intelligence which enables the inference of facts from rules by forward-chaining.

Descriptor

See attribute.

Development System

Functional component for the construction and maintenance of the application system.

Diagnosis

The target to be learned in a CBR system. For instance, the class attribute is a typical concept to be learned in an inductive system. However, other targets can be learnt within a CBR system: the justification of the solution, a problem solving path *etc.*

Domain Knowledge

See Background Knowledge

Evaluation Criteria

Guidelines for conducting a meaningful comparison between systems for different kinds of users, different levels of utilisation and different states of realisation.

Execution System

The result of using the development system on a particular application.

Expert System

Computer system that performs the task of an expert in a restricted area.

Forward-Chaining

Control procedure that produces new decisions by recursively applying the antecedent conditions of inferential rules.

Index

Attribute which narrows the search when looking for specific cases in a database.

Indexing Tree

Tree built on a set of indexes.

Induction

Method used in artificial intelligence which enables the learning of rules or concepts from a set of training examples.

Induction Tree

See decision tree.

Justification

Explanation of an action or a decision by presenting antecedent considerations, such as heuristic rules, that affected the motivation of making this action or providing this result.

Knowledge Acquisition

Extraction and formulation of knowledge from different sources. It should be the major goal of a learning system. Up to now, knowledge acquisition has to be performed mainly "by hand", especially with experts.

Knowledge Base

Name given to the conjunction of domain knowledge and a database that follows the domain requirements. Hence, the knowledge base is traditionally confounded with the database in statistics, where no additional knowledge is available, whereas it is confounded with the domain knowledge in pure knowledge-based systems like expert systems, where no cases are available.

Learning

The process of improving performance by acquiring or modifying knowledge.

Rule

A pair, composed of a condition and a consequence, that can be used in rule-based systems by deductive processes such as backward-chaining and forward-chaining.

Rule-Based System

Computer system that explicitly relies on a rule set.

Similarity Assessment

Computation of the similarity measure between two cases.

Similarity Measure

Mathematical function used to order the retrieved cases in the database, with respect to a given query. A similarity measure is called local when it is applied over specific attributes (numeric, symbolic, *etc.*); it is called global, when it summarises a set of local similarity measures.

Taxonomic Attribute

Attribute where the possible values are described in a hierarchy tree according to a specialisation relation. An example of a taxonomic attribute is the colour that may be described by two values (light and dark), each value may be specialised (dark is specialised by brown and black) *etc.*

Taxonomy

See taxonomic attribute.

Test Set

Part of the case base that is used to test a previously built system.

Training Set

Part of the case base that is used to build a system.