Case-Based Reasoning: A Review

Ian Watson & Farhi Marir

AI-CBR, Dept. of Computer Science, University of Auckland, New Zealand

Table of Contents

Abstract

Case-Based Reasoning (CBR) is a relatively recent problem solving technique that is attracting increasing attention. However, the numbers of people with first hand theoretical or practical experience of CBR is still small. The main objective of this review is to provide a comprehensive overview of the subject to people that are new to CBR. The paper will outline the development of CBR in the US in the 1980's. It will describe the fundamental techniques of CBR and contrast its approach to that of model-based reasoning systems. A critical review of currently available CBR software tools is followed by descriptions of CBR applications both from academic research and, in more detail, three CBR systems that are presently being used commercially. Each of the three commercial case studies highlights features that made CBR particularly suitable for the application. Moreover the last case study describes a development methodology for implementing CBR systems. The paper concludes with a research agenda for CBR. A detailed categorised bibliography of CBR research is provided in a companion paper to this [Marir & Watson, 94].

Introduction

"*reality takes shape in the memory alone*",
Marcel Proust, *A la recherché du temps perdu.*

Expert or knowledge-based systems (KBS) are one of the success stories of Artificial Intelligence (AI) research. In a recent survey the UK Department of Trade & Industry found over 2000 KBS in commercial operation (the survey excluded KBS in University research laboratories) [DTI, 92]. It has been around twenty years since the first documented KBS (the trinity of classic systems: DENDRAL, MYCIN and PROSPECTOR) were reported, yet in that time the basic architecture of KBS has changed little. The early KBS, and today's systems, are based upon an explicit model of the knowledge required to solve a problem - so called *second generation* systems [Clancey, 85] using a *deep* causal model that enables a system to reason using first principles. But whether the knowledge is shallow or deep an explicit model of the domain must still be elicited and implemented often in the form of rules or perhaps more recently as object models. However, despite the undoubted success of

model-based KBS in many sectors developers of these systems have met several problems:

- knowledge elicitation is a difficult process, often being referred to as the *knowledge elicitation bottleneck*;
- implementing KBS is a difficult process requiring special skills and often taking many man years;
- once implemented model-based KBS are often slow and are unable to access or manage large volumes of information; and
- once implemented they are difficult to maintain [Bachant & McDermot, 84; Coenen & Bench-Capon, 92; Watson et al., 92b].

Solutions to these problems have been sought through better elicitation techniques and tools [Motta et al., 89; Brooke & Jackson, 91], better KBS shells and environments, improved development methodologies [Diaper, 89; Inder & Filby, 1991; Watson et al., 92a; Wielinga et al., 1992], knowledge modelling languages and ontologies [Alexander et al., 86; Steels, 90; Chandraskaren 86 & 90; Wielinga et al., 92], facilitating the co-operation between KBS and databases in expert databases and deductive databases [Gallaire et al.,, 81; Minker, 88; Marir & Yip, 92; Marir 93], and techniques and tools for maintaining systems [Bench-Capon & Coenen, 92, Watson et al., 92b]

However, over the last few years an alternative reasoning paradigm and computational problem solving method has increasingly attracted more and more attention. Case-based reasoning (CBR) solves new problems by adapting previously successful solutions to similar problems. CBR is attracting attention because it seems to directly address the problems outlined above. Namely:

- CBR does not require an explicit domain model and so elicitation becomes a task of gathering case histories,
- implementation is reduced to identifying significant features that describe a case, an easier task than creating an explicit model,
- by applying database techniques largely volumes of information can be managed, and
- CBR systems can learn by acquiring new knowledge as cases thus making maintenance easier.

This paper therefore has several objectives.

- to introduce the history, techniques and application of CBR to a new audience,
- to provide a comprehensive review of CBR software tools,
- to demonstrate that the benefits of CBR are real and not merely hype, and to
- identify where further research and development is required.

To this end, section two presents a history of CBR from its foundations in the late seventies to the present day. Section three summarises CBR techniques, whilst section four provides a review of CBR software tools currently available. Section five describes influential CBR applications. These are divided into two categories: those that are primarily academic, implemented to demonstrate certain aspects of CBR, and three systems that are being used commercially. These systems are described in more detail than the academic demonstrators, highlighting why CBR was particularly suitable to the problem domain and describing the benefits it brought. The last case study describes how the system was implemented and outlines a methodology for developing CBR systems. Section six summarises by making a *case for case-based reasoning*, drawing on all previous sections. The paper concludes by identifying areas for further research. A companion paper to this presents a detailed categorised

bibliography of CBR for the researcher requiring a *reasonably* complete bibliography of the subject [Marir & Watson, 94].

Those interested in other overviews of case-based reasoning should refer to Riesbeck & Schank [89], Slade [91] and Kolodner [92 & 93] for comprehensive discussions of CBR concepts and origins. A task decomposition of CBR is provided by Aamodt & Plaza [94]. The review presented here differs from previous works in that in addition to describing the history and techniques of CBR and contrasting its approach with that of model-based reasoning systems, it also gives a critical review of currently available CBR software tools.

A History of Case-Based Reasoning

The work Schank and Abelson in 1977 is widely held to be the origins of CBR. They proposed that our general knowledge about situations is recorded as *scripts* that allow us to set up expectations and perform inferences. Scripts were proposed as a structure for conceptual memory describing information about stereotypical events such as, going to a restaurant or visiting a doctor. However, experiments on scripts showed that they were not a complete theory of memory representation - people often confused events that had similar scripts. For example, a person might mix up room scenes from a visit to a doctor's office with a visit to a dentist's office. Such observations fell in line with the theories of concept formation, problem solving and experiential learning within philosophy and psychology [Tulving, 77, Smith et al., 78].

Roger Schank continued to explored the role that the memory of previous situations (i.e., cases) and situation patterns or *memory organisation packets* (MOPs) play in both problem solving and learning [Schank, 82]. At a similar time Gentner [83] was developing a theoretical framework for analogy which also has relevance to CBR. Perhaps with the benefit of hindsight it is also possible to find references of significance to CBR in Wittgenstein's observation that *natural concepts* such as tables and chairs are in fact polymorphic and can not be classified by a single set of necessary and sufficient features but instead can be defined by a set of instances (i.e., cases) with family resemblances [Wittgenstein, 53]. This work has been cited by Aamodt and Plaza [94] as a philosophical basis for CBR.

Whilst the philosophical roots of CBR could perhaps be claimed by many what is not in doubt is that it was the work of Roger Schank's group at Yale University in the early eighties that produced both a cognitive model for CBR and the first CBR applications based upon this model. Janet Kolodner developed the first CBR system called CYRUS [Kolodner, 83a & b]. CYRUS contained knowledge, as cases, of the travels and meetings of ex-US Secretary-of-State Cyrus Vance. CYRUS was an implementation of Schank's dynamic memory model. Its case-memory model later served as the basis for several other CBR systems including MEDIATOR [Simpson, 85], CHEF [Hammond, 86], PERSUADER [Sycara, 87], CASEY [Koton, 89] and JULIA [Hinrichs, 92].

An alternative approach came from Bruce Porter's work, at The University of Texas in Austin, into heuristic classification and machine learning resulting in the PROTOS system [Porter & Bareiss, 86; Bareiss, 88]. PROTOS unified general domain knowledge and specific case knowledge into a single *case memory model*. This work was taken further by GREBE, a system operating in a legal domain [Branting, 91].

It is perhaps no surprise that since the practice of law is largely based upon precedence and the notion of cases, that there has been some interest from this sector in CBR. Notably Edwina Rissland's group at the University of Massachusetts in Amherst who developed HYPO [Ashley, 88]. In HYPO cases representing legal precedents are used to interpret a court situation and to produce arguments for both the defence and the prosecutors. This system was later combined with rule-based reasoning to produce CABARET [Rissland & Skalak, 89].

CBR research is not restricted to the US, but it was slower to get started in Europe. Amongst the first cited European work is that of Derek Sleeman's group from Aberdeen in Scotland. They studied the uses of cases for knowledge acquisition, developing the REFINER system [Sharma & Sleeman, 88]. At a similar time Mike Keane, from Trinity College Dublin, undertook cognitive science research into analogical reasoning that has subsequently influenced CBR [Keane, 88]

On continental Europe Michael Richter and Klaus Althoff [Althoff, 89], from the University of Kaiserslautern, applied CBR to complex diagnosis. This has given rise to the PATDEX system [Richter & Weiss, 91] and subsequently to the CBR tool S3-Case. Agnar Aamodt at the University of Trondheim has investigated the learning facet of CBR and the combination of cases and general domain knowledge resulting in CREEK [Aamodt, 89 & 91]

In the UK, CBR seems to be particularly applied to civil engineering. A group at the University of Salford are applying CBR techniques to fault diagnosis, repair and refurbishment of buildings [Watson & Abdullah, 94]. Yang & Robertson [94] in Edinburgh are developing a CBR system for interpreting building regulations, a domain reliant upon the concept of precedence. Whilst another group in Wales are applying CBR to the design of motorway bridges [Moore et al., 94].

Further afield there are active CBR groups in Israel [Oxman. 93a & b], India [Venkatamaran et al., 93] and Japan [Kitano, 93]. However, the increasing number of CBR papers in AI journals and the increasing number of commercially successful CBR applications is likely to ensure that many more countries take an active interest in CBR in the future. As an indicator the British Computer Society Specialist Group on Expert Systems has held CBR workshops suitable for novices at both it last annual conferences.

Case-Based Reasoning Techniques

A case-based reasoner solves new problems by adapting solutions that were used to solve old problems [Riesbeck & Schank, 89]

The CBR Cycle

The processes involved in CBR can be represented by a schematic cycle (see Figure 1). Aamodt and Plaza [94] have described CBR typically as a cyclical process comprising *the four REs*:

- RETRIEVE the most similar case(s);
- REUSE the case(s) to attempt to solve the problem;
- REVISE the proposed solution if necessary, and

- RETAIN the new solution as a part of a new case.

A new problem is matched against cases in the case base and one or more similar cases are *retrieved*. A solution suggested by the matching cases is then *reused* and tested for success. Unless the retrieved case is a close match the solution will probably have to be *revised* producing a new case that can be *retained*.

This cycle currently rarely occurs without human intervention. For example many CBR tools act primarily as case retrieval and reuse systems. Case revision (i.e., adaptation) often being undertaken by managers of the case base. However, it should not be viewed as weakness of CBR that it encourages human collaboration in decision support. The following sections will outline how each process in the cycle can be handled.

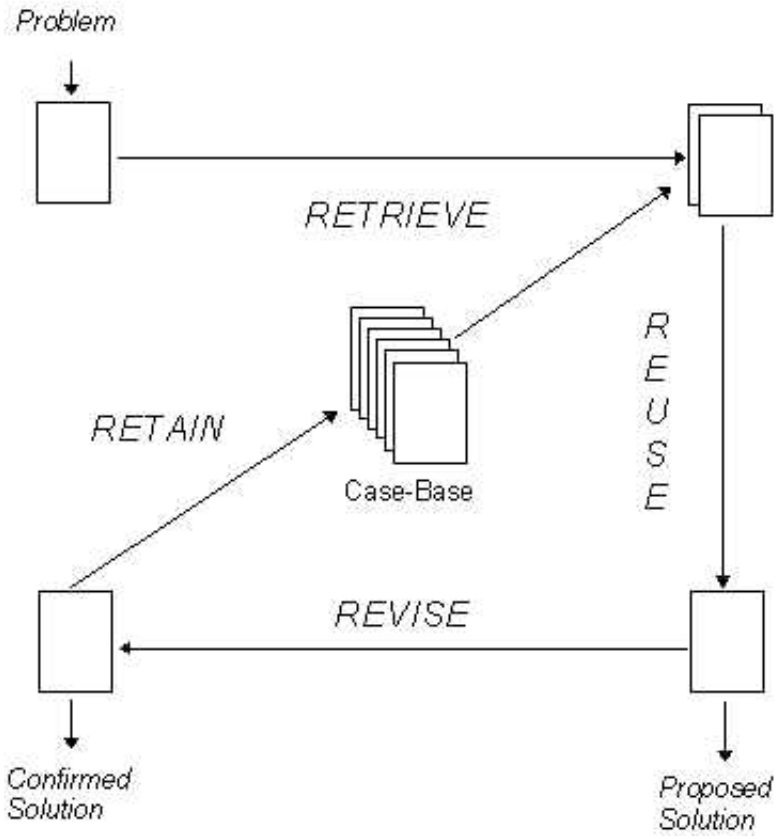

Figure 1 The CBR Cycle [adapted from Aamodt & Plaza, 1994]

Case Representation

A case is a contextualised piece of knowledge representing an experience. It contains the past lesson that is the content of the case and the context in which the lesson can be used [Alterman, 89, David, 91; Kolodner, 93]. Typically a case comprises:

- the *problem* that describes the state of the world when the case occurred,
- the *solution* which states the derived solution to that problem, and/or
- the *outcome* which describe the state of the world after the case occurred.

Cases which comprise problems and their solutions can be used to derive solutions to new problems, as in CASEY [Koton, 89]. Whereas cases comprising problems and outcomes can be used to evaluate new situations. If, in addition, such cases contain solutions they can be used to evaluate the outcome of proposed solutions and prevent potential problems as in MEDIATOR [Simpson, 85]. Cases can be represented in a variety of forms using the full range of AI representational formalisms including frames, objects, predicates, semantic nets and rules - the frame/object representation currently being used by the majority of CBR software.

There is a lack of consensus within the CBR community as to exactly what information should be in a case. However, two pragmatic measures can be taken into account in deciding what should be represented in cases: the functionality and the ease of acquisition of the information represented in the case [Kolodner, 93].

Indexing

Case indexing involves assigning indices to cases to facilitate their retrieval. Several guidelines on indexing have been proposed by CBR researchers [Birnbaum & Collins, 89; Hammond 1989). Indices should:

- be predictive,
- address the purposes the case will be used for,
- be abstract enough to allow for widening the future use of the case-base, and
- be concrete enough to be recognised in future

Both manual and automated methods have been used to select indices. Choosing indices manually involves deciding a case's purpose with respect to the aims of the reasoner and deciding under what circumstances the case will be useful.

There are an ever increasing number of automated indexing methods including:

- Indexing cases by features and dimensions that tend to be predictive across the entire domain i.e., descriptors of the case which are responsible for solving it or which influence its outcome (Acorn & Walden, 92). In this method the domain is analysed and the dimensions that tend to be important are computed. These are put in a checklist and all cases are indexed by their values along these dimensions. For example, MEDIATOR uses this method by indexing on type and function of disputed objects and relationship of disputants, whilst CHEF indexes on texture and taste. This kind of technique is called *checklist-based* indexing [Kolodner, 93].
- Difference-based indexing selects indices that differentiate a case from other cases as in CYRUS. During this process the system discovers which features of a case differentiate it from other similar cases, choosing as indices those features that differentiate cases best.
- Similarity and explanation-based generalisation methods, which produce an appropriate set of indices for abstract cases created from cases that share some common set of features, whilst the unshared features are used as indices to the original cases [Hammond, 87 & 89].
- Inductive learning methods [Lebowitz, 87], which identify predictive features that are then used as indices [Goodman, 89]. These techniques are widely used (e.g., in Cognitive system's ReMind) and commonly use variants of the ID3 algorithm used for rule induction.
- Explanation-based techniques, which determine relevant features for each case [Barletta & Mark, 88; Simoudis et al., 93]. This method analyses each case to find which of their features

are predictive ones. Cases are then indexed by those features.
- However, despite the success of many automated methods, Janet Kolodner believes that people tend to do better at choosing indices than algorithms, and therefore for practical applications indices should be chosen by hand [Kolodner 1993].

Storage

Case storage is an important aspect in designing efficient CBR systems in that, it should reflect the conceptual view of what is represented in the case and take into account the indices that characterise the case. The case-base should be organised into a manageable structure that supports efficient search and retrieval methods. A balance has to be found between storing methods that preserve the semantic richness of cases and their indices and methods that simplify the access and retrieval of relevant cases. These methods are usually referred to as *case memory models*. The two most influential case memory models are the *dynamic memory model* of Schank and Kolodner, and the *category-exemplar model* of Porter and Bareiss.

The dynamic memory model

The case memory model in this method is comprised of *memory organisation packets* or MOPs. MOPs are a form of frame and are the basic unit in dynamic memory. They can be used to represent knowledge about classes of events using two kind of MOPs:

- *instances* representing cases, events or objects, and
- *abstractions* representing generalised versions of instances or of other abstractions.

The case memory, in a dynamic memory model, is a hierarchical structure of *episodic memory organisation packets* (E-MOPs) [Kolodner, 83a & b], also referred to as *generalised episode* (GEs) [Koton, 89] developed from Schank's more general MOP theory [Schank, 82]. The basic idea is to organise specific cases which share similar properties under a more general structure (i.e., a generalised episode). A GE contains three different types of objects: *norms*, *cases* and *indices*. Norms are features common to all cases indexed under a GE. Indices are features which discriminate between a GE's cases. An index may point to a more specific generalised episode or to a case, and is composed of an index name and an index value.

The case-memory is a discrimination network where nodes are either a GE, an index name, index value or a case. Index name-value pairs point from a GE to another GE or case. The primary role of a GE is as an indexing structure for storing, matching and retrieval of cases. During case storage when a feature (i.e., index name and index value) of a new case matches a feature of an existing case a new GE is created. The two cases are then discriminated by indexing them under different indices below the new GE (assuming the cases are not identical). Thus, the memory is *dynamic* in that similar parts of two cases are dynamically generalised into a new GE, the cases being indexed under the GE by their differences.

However, this process can lead to a explosive growth in the number of indices as case numbers increase. So for practical purposes most CBR systems using this method limit the number of permissible indices to a limited vocabulary.

The category-exemplar model

This model organises cases based on the view that the real world should be defined extensionally with cases being referred to as *exemplars* [Porter & Bareiss, 86]. The case memory is a network structure of categories, semantic relations, cases and index pointers. Each case is associated with a category. Different case features are assigned different importance in describing a case's membership to a category. Three type of indices are provided, which may point to a case or a category:

- *feature links* that point from problem descriptors (features) to a case or category,
- *case links* that point from categories to its associated cases, and
- *difference links* pointing from categories to the neighbouring cases that only differ in a small number of features.

A feature is described by a name-value pair. A category's exemplars are stored according to their degree of prototypicality to the category. Within this memory organisation, the categories are inter-linked within a semantic network containing the features and intermediate states referred to by other terms. This network represents a background of general domain knowledge that enables explanatory support to some CBR tasks. A new case is stored by searching for a matching case and by establishing the relevant feature indices. If a case is found with only minor differences to the new case, the new case may not be retained, or the two cases may be merged.

Retrieval

Given a description of a problem, a retrieval algorithm, using the indices in the case-memory, should retrieve the most similar cases to the current problem or situation. The retrieval algorithm relies on the indices and the organisation of the memory to direct the search to potentially useful cases.

The issue of choosing the *best* matching case has been addressed by research into analogy [Falkenehainer et al, 86]. This approach involves using heuristics to constrain and direct the search. Several algorithms have been implemented to retrieve appropriate cases, for example: serial search [Navinchandar, 91; Acorn & Walden, 92; Simoudis et al., 93], hierarchical search [Maher & Zhang, 91] and simulated parallel search [Domeshek, 93].

Case-based reasoning will be ready for large scale problems only when retrieval algorithms are efficient at handling thousands of cases. Unlike database searches that target a specific value in a record, retrieval of cases from the case-base must be equipped with heuristics that perform partial matches, since in general there is no existing case that exactly matches the new case.

Among well known methods for case retrieval are: nearest neighbour, induction, knowledge guided induction and template retrieval. These methods can be used alone or combined into hybrid retrieval strategies.

Nearest neighbour

This approach involves the assessment of similarity between stored cases and the new input case, based on matching a weighted sum of features. The biggest problem here is to determine the weights of the features. The limitation of this approach include problems in converging on the correct solution and retrieval times. In general the use of this method leads to the retrieval time increasing linearly with the number of cases. Therefore this approach is more effective when the case base is relatively small. Several CBR implementation have used this method to retrieve matching cases, for example:

BROADWAY [Skalk, 92] for selection of car models, the Compaq SMART System [Acorn & Walden, 92] for a customer product support help desk, and ANON [Owens, 93] for situation assessment in plan failure.

A typical algorithm for calculating nearest neighbour matching is the one used by Cognitive Systems ReMind software reported in Kolodner [93] where *w* is the importance weighting of a feature (or slot), *sim* is the similarity function, and *fI* and *fR* are the values for feature *i* in the input and retrieved cases respectively.

$$\frac{\sum_{i=1}^{n} w_i \times sim\left(f_i^I, f_i^R\right)}{\sum_{i=1}^{n} w_i}$$

Figure 2 A Nearest Neighbour Algorithm

Induction

Induction algorithms (e.g. ID3 [Quinlan, 79]) determine which features do the best job in discriminating cases, and generate a decision tree type structure to organise the cases in memory. This approach is useful when a single case feature is required as a solution, and where that case feature is dependent upon others.

Knowledge guided induction

This method applies knowledge to the induction process by manually identifying case features that are known or thought to affect the primary case feature. This approach is frequently used in conjunction with other techniques, because the explanatory knowledge is not always readily available for large case bases.

Template retrieval

Similar to SQL-like queries, template retrieval returns all cases that fit within certain parameters. This technique is often used before other techniques, such as nearest neighbour, to limit the search space to a relevant section of the case-base.

Adaptation

Once a matching case is retrieved a CBR system should adapt the solution stored in the retrieved case to the needs of the current case. Adaptation looks for prominent differences between the retrieved case and the current case and then applies formulae or rules that take those differences into account when suggesting a solution. In general, there are two kinds of adaptation in CBR:

- *Structural adaptation*, in which adaptation rules are applied directly to the solution stored in cases [Kolodner, 93]. This kind of adaptation is used in JUDGE [Bain, 86] and CHEF [Hammond, 86].
- *Derivational adaptation*, that reuses the algorithms, methods or rules that generated the original

solution to produce a new solution to the current problem. In this method the planning sequence that constructed that original solution must be stored in memory along with the solution as in MEDIATOR [Simpson, 85]. Derivational adaptation, sometimes referred to a *reinstantiation*, can only be used for cases that are well understood.

An ideal set of adaptation rules must be strong enough to generate complete solutions from scratch, and an efficient CBR system may need both structural adaptation rules to adapt poorly understood solutions and derivational mechanisms to adapt solutions of cases that are well understood.

Several techniques, ranging from simple to complex, have been used in CBR for adaptation. These include:

- *Null adaptation,* a direct simple technique that applies whatever solution is retrieved to the current problem without adapting it. Null adaptation is useful for problems involving complex reasoning but with a simple solution. For example, when someone applies for a bank loan, after answering numerous questions the final answer is very simple: grant the loan, reject the loan, or refer the application.
- *Parameter adjustment,* a structural adaptation technique that compares specified parameters of the retrieved and current case to modify the solution in an appropriate direction. This technique is used in JUDGE [Bain, 86], which recommends a shorter sentence for a criminal where the crime was less violent.
- *Abstraction and respecialisation,* a general structural adaptation technique that is used in a basic way to achieve simple adaptations and in a complex way to generate novel, creative solutions. The PLEXUS planning system uses this technique [Alterman, 88].
- *Critic-based adaptation,* in which a critic looks for combinations of features that can cause a problem in a solution. Importantly, the critic is aware of repairs for these problems. PERSUADER [Sycara, 87] is a system which uses all the techniques of adaptation discussed above.
- *Reinstantiation,* is used to instantiate features of an old solution with new features. For example, CHEF can reinstantiate *chicken* and *snow peas* in a Chinese recipe with *beef* and *broccoli* thereby creating a new recipe.
- *Derivational replay,* is the process of using the method of deriving an old solution or solution piece to derive a solution in the new situation. For example, BOGART [Mostow et al. 1989], which replays stored design plans to solve problems.
- *Model-guided repair,* uses a causal model to guide adaptation as in CELIA [Redmond, 92], which is used for diagnosis and learning in auto mechanics, and KRITIK [Goel et al., 92] used in the design of physical devices.
- *Case-based substitution,* uses cases to suggest solution adaptation as in ACBARR [Moorman & Ram, 92, Ram et al., 93] a system for robot navigation

CBR Software Tools

Theoreticians might argue that the current surge in interest in CBR is due to the intuitive nature of CBR and because it may closely resemble human reasoning. Software vendors might argue that it is because CBR tools have made the theory practically feasible. There is truth in both views but certainly the tools have made a contribution. This section reviews most of the currently available major CBR

tools. In most cases these have been evaluated by the authors and this is indicated in the text. The section concludes with a brief description of other CBR tools that have been reported to be available.

CBR-Express

CBR-Express, produced by Inference Corporation, is perhaps the most successful CBR product to date. CBR-Express is specifically tailored to a vertical market, that of help desks. The customer support help desk is becoming an increasingly common feature in the nineties, and managers of such services face similar problems:

- for staff manning the help desk to be useful they require training,
- training takes time and money, and
- manning a help desk is not the best job in the world so staff turn over is high.

Where expertise is scarce conventional wisdom tells us a KBS is useful. However, as Dearden & Bridge [93] point out there is an additional problem. Customer help desks often deal with technical faults. To implement a model-based reasoning system (MBR) you need to know the faults and their solutions. But if you (as the manufacturer) new there was a fault you probably wouldn't release the product. Thus, help desks often must deal with faults that designers and engineers have not envisaged and consequently could not be in the model-base.

Dearden & Bridge argue that this is where CBR systems offer a solution. Through the application of the CBR cycle including case retainment (i.e., learning) newly identified faults and their solutions can be added as new cases as they arise (this is simpler than adding new rules). CBR-Express applies the CBR cycle very successfully to help desks and is currently the market leader for knowledge-based help desk software.

CBR-Express has a simple case structure and uses nearest neighbour matching to retrieve cases. A Sample case from CBR-Express that diagnoses faults in laser printers is shown in Figure 3. Developers using CBR-Express use an interface that deals with all programming elements of case creation and editing resulting in a syntax free environment that lets people without programming experience quickly develop case-bases [Watson & Abdullah, 94].

```
BEGIN CASE CASE11
TITLE
Ink cartridge is damaged, causing black stains.
DESCRIPTION
Stains appear as small, round, black dots that
occur on front or back of page.
Sometimes wide inconsistent stains appear.
QUESTIONS
Are you having print quality problems?
ANSWER : Yes
SCORING : (-)
What does the print quality look like?...
ANSWER : Black Stains
SCORING : (default)
Does cleaning the printer with cleaning paper
remove problem?
ANSWER : No
SCORING : (default)
ACTIONS
Check toner cartridge and replace if it is low in toner or damaged...
BROWSE TEXT
CREATION 29/7/91 14:19:22
LAST_UPDATE 29/7/91 14:19:22
LAST_USED 29/7/91 14:19:22
STATUS ACTIVE
```

END CASE

Figure 3 A CBR-Express Case

A key feature of CBR-Express is its ability to handle free-form text. This was felt to be vital to the help desk market since it lets customers describe their problems in their own words rather than being taken through a decision tree style question and answer session. CBR-Express ignores words such as: *and, or, I, there*, etc., it can use synonyms, and represents words as a set of *trigrams*. The trigram for *cartridge* is: CAR, ART, RTR, TRI, RID, IDG, DGE.

The use of trigrams means that CBR-Express is very tolerant of spelling mistakes and typing errors such as letter transpositions. The trigrams for *cartridge* and *cartrideg* will still match closely. Although there are obvious problems with this lexical approach it is nonetheless surprisingly powerful and very useful for CBR-Express' market.

CBR-Express examines a user's free form text entry and matches this against cases' *titles* and *descriptions*. This results in the retrieval of a set of cases. A list of ranked solutions with likelihood values is generated from the cases and the user is offered these along with a set of *questions*. Answers to these questions help narrow the number of cases that match leading to a more accurate solution that is presented to the user.

In the event of a solution not being reached (CBR-Express has a customisable threshold value) or if a solution is not satisfactory, CBR-Express closes the CBR cycle by using the concept of an *unresolved case*. If after searching on a problem description and answering several questions a successful match is not obtained an entire transcript of the consultation can be saved as an unresolved case. The administrator of the case base subsequently can find out what that case's solution was and modify the unresolved case to create a new case.

In summary, CBR-Express is extremely well suited to help desk applications and has also been used successfully for intelligent task assistance, information access systems and knowledge publishing. It is very easy to use, reliable, network ready, and notable for its intelligent handling of text. The product reviewed was CBR-Express Developers Kit version 1.2 R1 for Windows.

CasePoint

CasePoint, also from Inference Corp., is a runtime version of CBR-Express. CasePoint is a CBR delivery system as it only runs case-bases (i.e., users of CasePoint can not develop or edit cases, for this they must use CBR-Express) and does not contain the customer tracking facilities of CBR-Express. However, as a delivery vehicle it has certain advantages over CBR-Express. A criticism of nearest neighbour matching is that if a case base were large and if cases had many features it is not an efficient process. However, the matching algorithm that CasePoint uses is extremely fast.

CasePoint is also extremely memory efficient, only requiring a few hundred K to run. Moreover, since it is an efficient DDE server (MS Windows data exchange protocol) it can easily be integrated with other applications running under Windows. This is an interesting feature because although CBR-Express was designed for the help desk market, CasePoint can deliver CBR solutions for a wider range of diagnostic or classification problems at a reasonable cost. The product reviewed was CasePoint version 1.2.

ART**Enterprise*

ART**Enterprise* is the latest incarnation of Inference Corporation's flagship development product. In the 1980's ART and then ART-IM were marketed as AI development tools. Inference have dropped the AI label and now market ART**Enterprise* as "an integrated, object-oriented applications development tool. designed for MIS developers". ART**Enterprise* offers a variety of representational paradigms including:

- objects supporting multiple inheritance, encapsulation and polymorphism;
- rules; and
- cases.

This is all packaged with a GUI builder, version control facilities, and the ability to link to data repositories in most proprietary DBMS formats for developing client-server applications. Moreover, ART**Enterprise* offers cross-platform support for most operating systems, windowing systems and hardware platforms.

The CBR component in ART**Enterprise* is essentially the same as that in CBR-Express (or rather vice-versa since CBR-Express uses code from ART to provide its CBR functionality). Thus, ART**Enterprise* offers nearest neighbour matching and the impressive text handling mention above. However, the CBR functionality of ART**Enterprise* is more controllable than in CBR-Express. Moreover, the integration with other knowledge representational paradigms means that this offers an excellent environment to integrate CBR with other techniques and to use MBR techniques for case adaptation. However, although the package as a whole is very powerful the CBR functionality of ART**Enterprise* is less powerful than a tool such as ReMind.

ART**Enterprise* is currently (Spring 1994) undergoing advanced beta-testing with selected sites and will be available commercially shortly. At the time of publication the authors did not have first hand experience of this product. This review is based on information supplied by contacts within Inference, attendance at product seminars, and contacts with beta-testers.

Eclipse - The Easy Reasoner

Eclipse from Haley Enterprises is a close relative of ART. The forward chaining functionality of ART, written in LISP, was re-implement in C by NASA, entering the public domain as the language CLIPS. In the late eighties Paul Haley, the former Chief Scientist of Inference, developed a new ART-like language compatible with CLIPS. This became Eclipse.

Like ART, Eclipse offers objects, only this time fully compatible C++ objects, and optimised forward chaining using the Rete algorithm. Eclipse is available for the DOS operating system, MS Windows, UNIX platforms and certain mainframe environments. Haley Enterprise claim they can port Eclipse to any ANSI C environment in 30 days after you buy it.

The Easy Reasoner is a C library supplied with Eclipse that provides CBR functionality. The Easy Reasoner provides nearest neighbour and inductive retrieval of records in a database. Once records have been retrieved they can be asserted as Eclipse objects for adaptation by its rule-base. Eclipse supports the usual range of variable types and offers similar text handling facilities to ART (i.e., ignoring noise words and using trigrams to cope with spelling mistakes). Interestingly Eclipse also

uses *stems* to identify, for example, that *magnetically* and *magnetic* all stem from *magnet*.

Eclipse does not provide the same easy development environment as ART*Enterprise* and it lacks the ability to integrate data from disparate heterogeneous databases. But, for an experienced C++ programmer Eclipse may offer some advantages - it is extremely fast, and it can be embedded within a C++ application. The product reviewed was Eclipse version 1.1.

ReMind

ReMind, produced my Cognitive Systems Inc., was developed with support from the US DARPA programme. It was originally developed for the Macintosh and has since been ported to MS Windows and various UNIX platforms. It is available as a C library, for embedding in other applications, and as an interactive development environment. The product reviewed was the MS Windows version 1.1, and it must be said that the port from the Macintosh was all too obvious. However, despite problems with the interface ReMind is the most flexible of the CBR tools reviewed.

ReMind offers template, nearest neighbour, inductive, and knowledge-guided inductive retrieval. The template retrieval supports simple SQL-like queries returning all cases that fall within set parameters. The nearest neighbour retrieval is informed by user defined importance weightings that can be placed on case features. Inductive retrieval involves building a decision tree that indexes the cases. This can be done automatically by ReMind with no user involvement or the user can create a *qualitative model* to guide the induction algorithm.

Qualitative models are created graphically to indicate which concepts (case features) are dependent on other concepts. Qualitative weightings can be placed on these dependencies and ReMind then uses the qualitative model to guide the induction algorithm (hence *knowledge-guided induction*) resulting in decision trees that more closely reflect the causal relationship of concepts in the cases. Interestingly, different qualitative models can be created to explore different theories about the domain or to allow what-if questions to be asked.

Case adaptation is provided by creating adaptation formulae that adjust values based on the difference between the retrieved and the new case. These are also created graphically using a visual programming technique. Although this takes a little getting used to the extremely close typing of case features combined with the close typing of the operators does reduce syntax errors. Finally, ReMind can create case-bases from existing databases making it potentially useful for data-mining projects [Milne & Nelson, 94].

ReMind is a flexible CBR tool offering the widest range of case-retrieval methods along with interesting concepts such as qualitative models and visual adaptation formulae. It does not have the powerful text handling features of Inference's products and Eclipse, though it does provide an elementary natural language capability via a lexicon of terms that can be mapped to an ordered symbol hierarchy. However, in general users are forced to select rather than describe a situation.

Cognitive systems has announced CBR/text which will add significant new natural language and text handling functionality to ReMind. CBR/text is scheduled for release in the second quarter of 1994.

CASUEL

Ralph Bergmann, of the University of Kaiserslautern in Germany, reports that CASUEL, the Common Case Representation language developed by the INRECA project is now available (June 1994). INRECA (INtegrated REasoning from CAses) is a case-based reasoning project funded by the European Union under ESPRIT-contract P6322.

CASUEL is the interface language between all the INRECA component systems, but it is also intended to serve as the interface language between the INRECA integrated system and the external world, and as a standard for exchanging information between classification and diagnostic systems that use cases. Currently, CASUEL is not designed to handle design and planning tasks CASUEL is a flexible, object-oriented frame-like language for storing and exchanging descriptive models and case libraries in ASCII files. It is designed to model naturally the complexities of real cases. CASUEL represents domain objects in a class hierarchy using inheritance, slots being used to describe the objects, with typing constraints on slot values, as well as different kinds of relationships between objects.

In its current version, CASUEL also supports a rule formalism for exchanging case completion rules and case adaptation rules, as well as a mechanism for defining similarity measures. CASUEL is more concise than flat feature-value vectors for the representation of objects with a large number of potentially relevant attributes of different types, only a few of which are applicable to any given case. Its use reduces the number of information gain calculations needed for induction systems or similarity computations required for case-based reasoning.

Although CASUEL provides a lot of features, it does not require applications to use all of them. CASUEL is a keyword-driven language, that allows different system component to ignore irrelevant definitions. On the other hand, CASUEL is also open in the sense that new features can be defined if necessary for a particular kind of application or component.

CASUEL is intended to become the European standard for exchanging case-bases. It is planned to use CASUEL as a starting point from which a more general (including also planning and design tasks) Case Interchange Format will be developed in a joint European CBR-activity.

CASUEL Version 2.0 is now available in the public domain from the University of Kaiserslautern as a package including:

- a document describing the CASUEL V2.0 syntax,
- a set of case-bases in CASUEL V2.0 format, and
- a CASUEL V2.0 parser (written in C), which will be available soon.

Further information on CASUEL and details on how to download it can be obtained by email from casuel@informatik.uni-kl.de.

ReCall

ReCall is a CBR trademark of the Paris based AI company ISoft. This tool offers a combination of nearest neighbour and inductive case retrieval. ReCall is coded in C++ and is available on the PC under Windows 3.1, on UNIX Workstations under Motif, for: SUN, IBM RS6000, BULL DPX20, HP series 700, and DEC Alpha. It is designed on an open architecture allowing users to add CBR functionality to their own applications. The functionality in ReCall includes:

An object-oriented language with taxonomies, inheritance and multiple-inheritance mechanisms, typed descriptors, facets, deamons, and relationships between objects. This allows users to specify complex domain knowledge in a structured but modular way, and to describe cases having noisy, incomplete and uncertain descriptions.

Multiple hierarchical indices are used for organisation purposes and for efficient case retrieval. ReCall provides different methods for automatically analysing the case base providing for selection of indices as well as their organisation. However, application developers can impose their own organisation. Automatic procedures are based on inductive techniques. The automatic procedures takes into account the domain knowledge defined in the cases, helping users to develop applications interactively.

Similarity functions take into account both the properties and values of descriptors, as well as structural differences between cases. ReCall uses a variant of a nearest-neighbour algorithm that improves similarity computations.

ReCall supports two different adaptation mechanisms: a default adaptation mechanism based on a voting principal, and user defined adaptation rules. As ReCall is based on C++, external function calls can provide more complex adaptation.

ReCall has a graphical user interface that allows a user with no programming knowledge to quickly initialise a case-base. Through the use of specialised graphic editors, the user can define objects, relationships between objects, taxonomies, deamons and adaptation rules. A tree editor allows the user to interact directly on the case organisations in order to control and modify indices. A *user mode* allows developers to write simple adaptation rules or daemons easily, whilst a *developer mode* gives access to an interpreted language. Users design, modify and test applications interactively and can work on several case bases at the same time.

ReCall can be easily interfaced to external applications in particular with data bases and since ReCall is available as a C++ library, CBR functionality can be integrated with other applications. It has been used (according to ISoft) for:

- diagnosis, including: user-help systems (hot-lines), failure diagnosis, and maintenance help systems;
- design systems, including: cost estimation, document layout and application sizing;
- simulation, including: estimation, prediction, market forecasts, and behavioural simulation;
- control and monitoring, including: telecommunication network supervision, alarm filtering, situation evaluations, crisis situation management, performance analysis, and continuous process control; and for
- information retrieval, including: query analysis, and DBMS optimisation.

Information about ReCall was provided by ISoft.

Other CBR Tools

AI software companies tend to come and go, but at the time of writing the following tools with a CBR component have been reported. This is not necessarily a complete list, and the authors have no first hand experience of these products.

Esteem, from Esteem Software Inc., was originally written in Intellicorp's Kappa-PC. It has been re-written in C++ and has its own inference engine enabling developers to create adaptation rules. It supports case hierarchies that help narrow the search. It also supports applications that access multiple case-bases and nested cases. This means that one can reference another case-base through an attribute slot in a case. Esteem also provides hierarchical retrieval allowing control of the induction process through feature counting, weighted feature computation and inferred computation.

CasePower (formerly called Induce-it) from Inductive Solutions Inc. builds its cases within the matrix environment provided by Microsoft Excel, a spreadsheet application. Rows and columns of cells within a spreadsheet are used to define cases and their attributes - cells containing attribute values. Associated worksheets and formulas incorporated as macros are used to provide the CBR algorithms and the weights associated with values. All the tools described previously rely on an object/frame representation to model cases. Also many tools (e.g., ART*Enterprise*, ReMind and ReCall) have rules or formulae and an inference engine to facilitate case adaptation. In contrast, CasePower is a specialised tool for constructing Excel spreadsheets that model case-based KBS. Within the limited confines of Excel it provides basic CBR functionality suitable for numeric applications . However, for more complex non-numerical applications another CBR tool may be preferable

KATE, produced by Acknosoft in Paris, is a KBS development environment with CBR functionality. CaseWork, KATE's CBR component integrates an instance based CBR approach with a tool for inductive learning and problem solving through the creation of decision trees. The inductive and case-based methods can be used separately or as a combined method. There are graphical editing facilities to build the case-index structure and to generate dialogues. KATE has incorporated initial results on the integration of case-based and inductive methods from the INRECA project.

S3-Case, derived from PATDEX, is part of the German company TechInno's S3 environment for systems maintenance.

CBR Applications

This section describes CBR applications. It is divided into two sub-sections. The first describes CBR applications that are the primarily the product of academic research. These system, mostly developed in the US, demonstrate certain features of CBR. They should be viewed as demonstrators, as there is little evidence they have been used in real situations.

Although CBR is a relatively new method there have been several successful commercial applications. The second sub-section discusses three such applications. The first developed at Lockheed is usually cited as the first commercial CBR application. Lockheed's CLAVIER system has already become the *classic* CBR system demonstrating:

- that CBR could be applied to solve a problem where no explicit model existed, and
- that CBR systems can learn by acquiring new cases and so improve their performance with time.

The second developed by British Airways shows how CBR techniques are easily accepted by users because of CBR's intuitive feel and how retaining the rich context of a case has advantages over the distillated knowledge associated with rule-based systems. The final system developed for Legal &

General shows how CBR systems can be implemented rapidly and can be maintained by users.

Academic Demonstrators

In the relatively short time since the first CBR systems were implemented there have been numerous academic CBR demonstrators. This sub-section presents a selection of the better known classified according to tasks (where a system could be placed in several categories it has only been placed in one). This section is not exhaustive, but is intended to give practitioners an indication of the range of problems to which CBR has been applied. A more complete description of CBR demonstrators is given in Kolodner [93].

Knowledge acquisition

The REFINER program [Sharma & Sleeman, 88] is a knowledge acquisition system that helps an expert refine his knowledge in a more natural way than extracting rules from an expert. The system has the ability to recognise classifications that are inconsistent and to suggests ways of resolving the inconsistency. REFINER uses both inferred rules as well as individual cases.

Legal reasoning

JUDGE [Bain, 86] represents a case-based model of criminal sentencing. The program starts with a simple set of strategies for forming sentences and then begins to retrieve remindings of its own cases for developing new sentences. JUDGE reasons about murder, man-slaughter and assault cases. JUDGE has five stages of operations which are: interpretation, retrieval, differential analysis, application and modification and generalisation. JUDGE uses its case-base to maintain a consistent sentencing pattern.

HYPO [Ashley, 88] does case-based legal reasoning in the area of patent law. Centred on cases claiming violation of a patent, such as the release of a trade secret, HYPO uses its case-base of precedents to generate plausible arguments for the prosecution and the defence.

GREBE (Generator of Recursive Exemplar-Based Explanation) [Branting, 91] uses knowledge in form of generalisations and category exemplars to determine the classification of new cases. GREBE's knowledge-base contains rules and a small collection of exemplar cases concerned with Texas' laws for injuries to workers travelling outside of the work place. GREBE uses a semantic network representation for cases and uses generalisation and exemplar-based explanation. GREBE represents an advance over previous exemplar-based systems in that it accepts detailed semantic network representations of cases, retains and reuses the explanations of category exemplars and use exemplar based reasoning recursively to assess similarity.

KICS [Yang & Robertson, 94] is being implemented in the domain of building regulations. This system accumulates case histories of interpretation of regulations used to establish precedents. These precedents can be used when revising statutory regulation and enrich the resulting new versions of regulations. They also provide relevant information for the experts to interpret and make decisions about cases coming to appeal.

Explanation of anomalies

SWALE [Kass, 86] is a case-based creative explainer. It has a library of cases for explaining why

animals and people die. If SWALE is given an anomalous event such as the death of a healthy, young horse, it searches for explanations of death in other context such as rock stars dying from drug overdoses or a spouse being murdered for life insurance, and then SWALE tries to adapt those explanations to fit the current situation.

Diagnosis

PROTOS [Porter & Bareiss, 86] was developed in the domain of clinical audiology. It learned to classify hearing disorders from descriptions of patients' symptoms, history and test results. The design of PROTOS was motivated by real-world constraints on knowledge representation (e.g., the limited polymorphy of concept instances), and learning (e.g., the limited availability of training). PROTOS was trained with 200 cases in 24 categories from a speech and hearing clinic. After training PROTOS had absolute accuracy of one hundred per cent.

CASEY [Koton, 89] is a system which diagnosis heart failure. As input it uses a patient's symptoms and produces a causal network of possible internal states that could lead to those symptoms. When a new case arises CASEY tries to find cases of patients with similar but not necessarily identical symptoms. If the new case matches, then CASEY adapts the retrieved diagnosis by considering differences in symptoms between the old and new cases.

CASCADE [Simoudis, 92] is a system for diagnosing the causes of crashes to the VMS operating system to suggest a solution. Although this system is purely a *retrieve and suggest* system (i.e., there is no adaptation) it assists effective recovery from a system crash.

PAKAR [Watson & Abdullah, 94] is a system that identifies possible causes for building defects and suggests remedial actions. Developed in CBR-Express under MS Windows PAKAR is able to combine textual information and computer aided design drawings to advise on possible causes for defects and potential solutions.

Arbitration

MEDIATOR [Simpson, 85] works in the domain of dispute resolution. Given a conflict between several parties, it proposes possible compromises. If one proposal fails to satisfy all the parties involved, it generates new proposals and records the failure thus avoiding a similar failure in the future.

PERSUADER [Sycara, 87] proposes resolution of disputes between labour and management. The input is a description of a dispute between management and labour over wage-benefit packages and the output is a compromise package adapting contracts that have been used by similar companies. The existence of a backup planner makes it unlike other CBR systems since it can generate new contracts when no existing ones can be found or adapted.

Design

CYCLOPS [Navichandra, 89] is a design problem solver. It solves design problems in the domain of landscaping. It is focused around finding potential problems when designing new amenities.

JULIA [Hinrichs, 92] is a case-based designer that works in the domain of meal planning. It uses cases

to propose solutions, decomposing the problem as necessary and posting constraints to guide synthesis. It exploits a repertoire of adaptation methods to transform previous dishes in order to meet the constraints of the current problem. These adaptation methods are used both to modify previous cases and repair previous decisions that have been invalidated by constraints that arrive late.

CADET [Sycara, 92; Sycara & Navichandra, 92] is a case-based design system that functions as designer's assistant for mechanical design. It retrieves previous successful designs while avoiding previous failures such as poor materials or high costs. CADET transforms abstract descriptions of the desired behaviour of the device into a description that can be used to retrieve relevant designs and generate a variety of equivalent alternative designs for a given set of design specifications.

ARCHIE [Pearce et al., 92] is implemented using ReMind and helps architects in the high-level task of conceptual design. It gives architects access to office building designs created by other architects and points to factors that should be considered in solving a given problem. Each case in ARCHIE contains several types of information Including: design goals, design plans, descriptions of how well the plan satisfied its goals and constraints and finally lessons to be learned from the case.

Planning

BATTLE [Goodman, 89] projects the results for plan evaluation in the domain of land warfare planning. The system was built from an existing database of 600 cases. The user describe a battle situation and chosen battle plan, BATTLE retrieve cases that are composed of pieces of battles and experts' evaluations, to project the outcome.

BOLERO [Lopez & Plaza 93] builds a diagnostic plan according to information known about a patient. The BOLERO-RBS system is a combination of a case-based reasoning and a rule-based system. BOLERO plays the role of a meta-level reasoner, whilst the RBS plays the role of the object-level reasoner. The RBS has knowledge about specific diseases in order to solve problems, while BOLERO has planning knowledge particular to that disease

TOTLEC [Costas & Kashyap, 93] was developed to solve complex manufacturing planning problems such as the detection of errors during the design phase, warning and advising the user about non-manufacturable designs. Its cases are stored in dynamic memory organisation. It uses a hierarchical, case-based planning paradigm, a complex indexing of cases, and a three stage incremental retrieval of cases based on the notion of similarity.

Repair and adaptation

CHEF [Hammond, 86] creates new recipes from old ones. CHEF begins planning by finding a recipe that satisfies as many of its active goals as possible. It uses a set of object critics and modification rules to change the old recipe and satisfy the goal of the new one. One of the important aspects of CHEF is explanation of failures through a causal description of why they occurred. CHEF link recipes to a failure's explanation to predict the failure in similar circumstances. CHEF stores new recipes indexed by the goals that they satisfy and the problems that they avoid.

PLEXUS [Alterman, 86] is a planner that adapts old plans to new situations. This system is designed to compare two plans by adapting the normal plan for riding San Francisco's subway system into a plan for riding New York's subway system. It has a mechanism for comparing the similarity, and

dissimilarity of two cases in the adaptation process.

COACH [Collins, 87] is a program which works in a football domain, COACH generates new football plays by improving old plays by debugging and repairing stored plans. COACH has relatively few plays in its library, but it has several strategies for adaptation to form new plays.

Tutoring

DECIDER [Farrel, 87] is a system that helps students understand or resolve a pedagogical problem by selecting and presenting appropriate cases from a database that respond to the student's goal.

Another case-based system tutoring system is described in [Aleven & Ashley, 92]. The system is used to generate fresh cases for analysis in response to a particular issue of interest as identified by a tutor. In this system the tutor enters as input the theme he wants to teach and the system presents a set of pedagogically related cases that are used by the tutor to teach that particular theme to students.

Commercial Applications

Leake [94] reports that at a recent workshop Janet Kolodner identified over 100 fielded CBR applications. This sub-section describes three commercial applications in some detail.

Lockheed - CLAVIER

The first commercially fielded CBR application was at Lockheed, Palo Alto [Hennessy & Hinkle, 92]. Modern aircraft contain many elements that are made up from composite materials. These materials require curing in large autoclaves. Lockheed, the US aerospace company, produce many such parts. Each part has its own heating characteristics and must be cured correctly. If curing is not correct the part will have to be discarded. Unfortunately, the autoclave's heating characteristics are not fully understood (i.e., there is no model that operators can draw upon). This is complicated by the fact that many parts are fired together in a single large autoclave and the parts interact to alter the heating and cooling characteristics of the autoclave.

Operators of Lockheed's autoclaves relied upon drawings of previous successful parts layouts to inform how to layout the autoclave. However, this was complicated by the fact that layouts were never identical because parts were required at different times and because the design of the composite materials was constantly changing. Consequently operators had to select a successful layout they thought closely matched and adapt it to the current situation.

This closely resembled the CBR paradigm and when Lockheed decided to implement a KBS to assist the autoclave operators they decided upon CBR. Their objectives were to:

- reuse previously successful loadings,
- reduce the pressure of work on one or two experts,
- secure the expertise of the experts as a corporate asset, and
- help to train new personnel.

The development of CLAVIER started in 1987, and it has been in regular use since the Autumn of 1990. CLAVIER searches a library of previously successful autoclave layouts. Each layout is

described in terms of:

- parts and their relative positions on a table
- tables, and their relative positions in the autoclave, and
- production statistics such as start and finish times, pressure and temperature.

CLAVIER finds substitutes for parts in a layout that do not match, and it recommends new layouts to operators. In adapting new layouts from previous ones CLAVIER:

- creates new layouts by adapting pieces of previous layouts
- minimises the number of required parts not included in the layout,
- maximises the number of high-priority parts included in the layout, and
- maximises the total number of parts in a layout.

CLAVIER acts as a collective memory for Lockheed and as such provides a uniquely useful way of transferring expertise between autoclave operatives. In particular the use of CBR made the initial knowledge acquisition for the system easier. Indeed, it is doubtful if it would have been possible to develop a MBR system since operatives could not say *why* a particular autoclave layout was successful. CLAVIER also demonstrates the ability of CBR systems to learn. The system has grown from 20 to over 150 successful layouts and its performance has improved such that it now retrieves or adapts a successful autoclave layout 90% of the time.

British Airways - CaseLine

CaseLine is a first generation technology demonstrator used by British Airways (BA) to assess the potential of CBR [Magaldi, 94]. CaseLine assists Boeing 747-400 technical support engineers with aircraft fault diagnosis and repair between aircraft arrival and departure. It advises on past defects and known successful recovery and repair procedures. When a fault in a Boeing 747-400 is detected or suspected, either by monitoring equipment or the pilots, details are transmitted to ground staff. The plane may only be scheduled to be on the runway for one hour during which time engineers have to identify the cause of the fault and effect repair. This is complicated because defects are often obscure and have complex and inconsistent causes. To delay the plane will disrupt schedules and costs thousands or pounds per minute. To let the plane take off with an unresolved fault could have catastrophic consequences

CaseLine is implemented in ReMind. Users can input diagnostic information and control the search for available repair and recovery information. The system contains around 200 cases (early 1994) that describe previous failure instances and details of successful recovery actions.

Three main search modes are provided:

1. *ATA Chapter* - a simple two digit number referring to a fault in the plane's maintenance manual,
2. *EICAS Message* - a precise but variable length alphanumeric text indicating a fault, and
3. *Reported Defect* - a variable length string describing a fault.

These can be used alone or together for case retrieval using either nearest neighbour or inductive retrieval. Usually, a single case is retrieved if an *exact* match has been specified or several cases if a *partial* match was required. CaseLine helps engineers identify procedures that have the highest

likelihood of success. The engineer is still obliged to use the aircraft maintenance manuals as a final authority and to follow approved procedures. But CaseLine does reduce costly delays by cutting out less productive routes to fault analysis and fault finding.

Initial assessment by BA states that "CBR has a set of in-built capabilities that complement a specific range of engineering problems. These are by nature, often more than just technical in origin, and require an awareness of many competing human, organisational and operational factors when posing solutions" [Magaldi, 94]. In particular BA identify three benefits of CBR:

- CBR is intuitive to both developers and users,
- CBR complements human reasoning and problem solving, and
- CBR retains the rich context of a problem situation - they discard nothing but simply index on different features of what they store.

The latter point is of particular legal interest. If an rule-based diagnostic system were developed its rules would represent distillated knowledge. The original reasons why a rule was created may become obscured with time. However, episodic cases are always heavily contextualised. If BA were sued for negligence using CaseLine they could demonstrate that engineers had followed procedures that had proved successful in case X - a simple defence. However, if BA used a rule-based system expert witnesses might have to prove that a rule was *theoretically* correct - a more complex defence.

Legal & General - SWIFT

Legal & General (L&G) are a major UK provider of financial services. Their IT department has an annual budget of around £60 million. In 1993 the company was in the process of down-sizing from dumb terminals attached to mainframes to PC based LANs. As part of a business process reengineering project they wanted to provide a streamlined service to employees purchasing PC's, peripherals, software or upgrades.

The system was developed very rapidly using a KBS development methodology called the Client Centred Approach (CCA). The CCA combines the linear stages of the Waterfall approach to software development with the iterative prototyping methods popular with KBS developers. The CCA explicitly encourages the involvement of all stakeholders in a project and emphasises the need to consider maintaining the system [Watson et al., 92a&b]. The CCA has seven deliverables:

- a feasibility study that identifies stakeholders, benefits, costs and resources;
- a concept system that illustrates the high level functionality of the system - useful for obtaining resource commitment;
- a demonstrator that proves the system is technically achievable;
- a trustable system where the case-base is sufficiently complete to be useful;
- a usable system that addresses interface and integration issues;
- a saleable system that can be *sold* to end-users or other organisational units; and
- a system embedded-in-use that has an ongoing managed maintenance plan.

A week was spent in April 1993 analysing the existing process and reengineering it. This resulted in a very much simplified design whereby all L&G's employees would have access to a single point of contact for ordering PC products and upgrades. An essential component of the reengineered process was a KBS that would contain knowledge about L&G's IT strategy, their approved product range and

the hardware/software options that different business units used.

Because of the volatile nature of the PC market it was decided that it would be important to make the maintenance of the KBS as easy as possible. It was decided that the managers of the service ideally should be able to maintain the knowledge-base themselves. Consequently, CBR was chosen as the knowledge representation paradigm that would best meet these constraints [Vargas & Raj, 93]. Inference's CBR-Express/CasePoint combination, Asymetrix Toolbook and Microsoft's Access were chosen to develop the demonstration system. After approximately one months prototyping a demonstration system, called SWIFT, was shown at several seminars to stakeholders from all the business units within L&G.

These presentations were carefully organised as part of a comprehensive communications plan. They were professionally conducted and involved describing why the existing process had to be reengineered, what benefits the new process would deliver, and concluded with a demonstration of the CBR software supporting the process. These sessions were essential in obtaining the support of the whole of the company to the project.

Following this an intensive system development stage took place that involved establishing the required databases and obtaining cases from different business units. In general at this stage prototypical cases were constructed from elicitation meetings held with representatives (usually IT specialists) from each business unit.

At the beginning of September 1993, less than five months after the project started, the system went live. L&G had decided to roll the system out incrementally:

- first it would be used by selected people on selected projects that would be carefully monitored enabling bugs, technical difficulties, organisational and communication problems and simple oversights to be solved;
- then two weeks later it would be rolled out to serve the whole IT department; and finally
- it would be rolled out business unit by business unit over six to nine months.

This phased roll out was crucial since it allowed problems to be trapped early and avoided many of the problems associated with a *big bang* approach to delivering a new system. By Christmas of 1993 SWIFT was operational across over two thirds of L&G's business units and was judged a success.

SWIFT operates by:

1. Obtaining a customer's business unit and location from their employee ID number using a relational database.
2. Asking if the enquiry is about software, hardware or upgrades and opening the relevant case-base.
3. Obtaining a free text query such as "*I want to buy a fast modem that lets me send faxes*".
4. Using that text to retrieve a set of matching cases.
5. Asking several questions to confirm which case matches best.
6. Offering a solution.

If the customer accepts the solution SWIFT enters the product ordering process. However, if the customer rejects the solution a full transcript of the consultation is stored as an *unresolved case*. At a

monthly meeting managers of the service study unresolved cases to determine if cases need adapting or if new cases are required. Recognising that knowledge changes is particularly important in the domain of personal computing where new products are brought out almost daily.

The L&G SWIFT system demonstrates that:

- CBR systems can be developed quickly,
- CBR systems can be effectively integrated within a wider information system, and
- that once established a CBR system can be maintained by people who are not programmers.

The CCA development methodology, which was developed to assist prototyping model-based KBS, proved valuable in guiding the development of SWIFT. In particular, the emphasis that the CCA places on involving stakeholders in the development process and the explicit attention to system maintenance from the outset fitted the CBR paradigm well. Moreover, the ability to rapid prototype the system within the constraints of linear milestones gave developers flexibility and project managers control.

The Case for Case-Based Reasoning

This section discusses the problems associated with developing model-based knowledge-based systems. It posits that CBR appears to offer solutions to many of these problems, and presents evidence from the literature to support these claims.

As was mentioned in the introduction, during the last thirty years many KBS have been developed that have an explicit model of the problem domain in which they operate. In many such systems the model is implemented by rules, and perhaps more recently by objects. In *second generation* systems [Clancey 1985] a *deep* underlying causal model exists that enables the system to reason from first principles in its application domain. There is little doubt that such MBR systems (whether they be deep or shallow) can be very successful. However, as identified in section 1 there are five major problems with this approach:

- knowledge elicitation is difficult
- KBS can be very complex and can take many man years to develop,
- KBS are frequently slow,
- KBS are often poor at managing large volumes of information, and
- once developed they are difficult to maintain.

The first problem was recognised as soon as KBS were built and was often attributed to the *knowledge elicitation bottleneck* [Hayes-Roth et al., 1983]. The second problem is familiar to any KBS developer and has partially been responsible for the increasing interest in KBS development methodologies and of knowledge modelling languages and ontologies. The third problem has partially been overcome by the ever decreasing cost of processing power, whilst solutions to the fourth have been sought through the integration of AI techniques with database technology. However, for many years practitioners believed that KBS were easy to maintain - almost all books on KBS development written during the eighties will contain a quote similar to "maintaining a rule-base is easy, being simply a matter of adding or subtracting rules from the knowledge-base" Easier than maintaining procedural C or FORTRAN code true, but not *easy*. Unfortunately, the experience of XCON/R1 [Bachant &

McDermot, 84] and others [Coenen, & Bench-Capon, 92; Vargas & Raj, 93] has shown that maintaining model-based KBS is not as simple as adding or subtracted rules or objects. As a knowledge-base grows it becomes a complex debugging task.

However, there is a more fundamental problem that has been overlooked. KBS practitioners did not consider how to build a KBS when there was no model available. Overlooking this problem reflects the heritage of KBS in academic research laboratories. The early KBS (e.g., DENDRAL, MYCIN, PROSPECTOR) all operated in domains where there were good underlying models (either from first principles or statistical) - scientists are comfortable with working with models, they build them for a living. Unfortunately, in a commercial environment and outside of the Universities many people make decisions without reference to first principles and underlying causal or statistical models.

These people solve problems by using their experience. It is no surprise that *expert* and *experience* derive from the same root. We posit that the KBS community was seduced by rules and neglected the truism that experts solve problems by applying their experience, whilst only novices attempt to solve problems by applying rules they have recently acquired. The application of experience to problem solving is the hallmark of CBR. Thus, CBR is proposed by some as a psychological theory of human cognition [Slade, 91] and one that provides a cognitive model of how people solve problems [Kolodner, 91]. It offers a paradigm that is claimed to be close to the way people solve problems and one that overcomes the brittleness of MBR systems [Barletta, 91; Helton, 91]

Hence, there is a strong case for CBR since it has several potential advantages over MBR:

- CBR systems can be built without passing through the knowledge elicitation bottleneck since elicitation becomes a simpler task of acquiring past cases. This was demonstrated by the CLAVIER system.
- CBR systems can be built where a model does not exist, this is also well demonstrated by the CLAVIER system.
- Implementation becomes a simpler task of identifying relevant case features, and moreover a system can be rolled out with only a partial case-base as happened with CLAVIER, CaseLine and SWIFT. Indeed, using CBR a system need never be *complete* since it will be continually growing. This removes one of the bug-bears of KBS - how to tell when a knowledge-base is complete.
- CBR systems can propose a solution quickly by avoiding the need to infer an answer from first principles each time, important in CaseLine and in most help-desk situations.
- Individual or generalised cases can be used to provide explanation that are perhaps more satisfactory than explanations generated by chains of rules, important in many domains with legal implications as with CaseLine.
- CBR systems can learn by acquiring new cases making maintenance easier as demonstrated by CLAVIER, CaseLine and SWIFT.
- Finally, by acquiring new episodic cases CBR systems can grow to reflect their organisation's experience. If a rule-based KBS were delivered to six companies and used for six months, after that time each system would be identical, assuming no maintenance had taken place. If six identical CBR systems were used in a similar way after six months there could be six *different* systems as each could have acquired different episodic cases.

The claim that CBR systems can be implemented faster than MBR systems was supported by a study conducted by Cognitive Systems which stated that it took two weeks to develop a case-based version

of a system that took four months to build in rule-based form [Goodman 89]. Also, and more recently, developers at Digital Equipment Corporation confirmed that a rule-based system called CANASTA took more than eight times longer to develop than CASCADE a case-based system with the same functionality [Simoudis, 92; Simoudis et al., 93]. They also claim that the maintenance of CANASTA is continual whereas CASCADE needs almost no maintenance. Related claims are provided by Hennessy and Hinkle [92] concerning CLAVIER and Vargas & Raj [93]. However, isolated claims such as these should be treated with caution lest CBR is hyped in the same way the knowledge-based systems were a decade or more ago. We should also not overlook the fact that for well understood domains MBR systems can be very effective and are now a relatively mature technology.

Conclusions

The US is still the theoretical and practical centre of case-based reasoning and CBR continues to be put forward as a model for human memory and reasoning. Although there is currently limited practical evidence, there is anecdotal evidence from knowledge engineers building CBR systems who state that experts and users seem more comfortable with CBR than MBR [Vargas & Raj, 93; Leake, 94; Magaldi, 94].

Thus, CBR does seem to address many of the shortcomings of MBR, particularly the knowledge elicitation bottleneck, the ability of the systems to learn incrementally and to provide highly contextualised explanations. This is why over the last few years, this new paradigm for building KBS has grown from an isolated research area to a field of widespread interest. Originally, it was an area of research in AI that was exclusively the focus of academic research and was little known by commercial AI developers. However, CBR emerged from the research labs as the new hot topic both for industry and academe with the development of commercial products addressing this area of reasoning [Harmon, 92].

These developments have produced commercial CBR tools which use database and AI techniques that can help sift through large amounts of historical information in order to automate knowledge based on experience. This automation of historical experience required reasoning about previous cases which could complement other methods of problem solving. It is extremely natural, for example, to be reminded of something when trying to solve a particular problem. The goal of CBR systems is thus to be reminded of similar cases when faced with a new problem and to retrieve those cases in order to help the user solve the new problem in a fashion similar to the way it was solved for a closely matching situation.

With all of its advantages in domains that are poorly understood there are still problems that need to be worked out for CBR. If for example, MBR systems face problems in eliciting, acquiring, representing and maintaining knowledge, there are also sensitive problems that face CBR including:

- how cases should be represented,
- how indices should be chosen for organising memory efficiently,
- how to structure relationships between cases and parts of different cases,
- how to handle cases containing multimedia,
- how to handle massive case-bases, and
- how to develop general adaptation heuristics for modifying previous cases or their solutions to

fit new cases.

In the future CBR systems may need to be able:

- to forget unused cases in order to maintain case-base efficiency, and
- learn about the emergence of any indices that had not previously been thought significant.

More immediately it is necessary to integrate CBR fully with other reasoning paradigms and information systems and importantly to critically evaluate the effectiveness of CBR in commercial domains. Relying on previous experience without validation may result in inefficient or incorrect solutions being recommended causing an increase in problem-solving time or errors that may have negative effects on the process of learning.

CBR is entering a period where people are exploring its potential and its limitations. This is largely being driven by the available tools and we would be unwise to neglect the theoretical basis of CBR. In the next years we will see many more hopefully successful applications using CBR. The majority of these will undoubtedly be in help desks. However, despite the *shallow* nature of many of these systems, the AI community should be proud that after all the hype surrounding KBS we are finally able to deliver a simple effective solution that business can understand and benefit from. Moreover, the acceptance of these systems, of CBR and thus by implication of AI techniques in general may open the door to a more widespread use of AI and its integration with management information systems. Inference's new tool ART**Enterprise* seems to be in the vanguard of corporate AI and will encourage the spread of cases as a corporate resource and of knowledge as an asset to be managed.

Of all the challenges identified above perhaps the greatest is adaptation. Relatively few CBR systems adapt cases, and those that do usually do so in a simplistic manner. Many CBR systems, like CBR-Express, are primarily case retrieval systems, adaptation being left to human intervention. Perhaps the other great challenge will be handling massive case-bases. Currently most CBR systems are relatively small, hundreds not thousands of cases. We have little theoretical or practical understanding of how CBR will perform where cases are feature rich and where there are many thousands of them. Brute force solutions such as the application of massively parallel computation have been proposed [Kitano, 93], but are unlikely to be commercially relevant for some time.

Despite these problems CBR will continue to be applied to a wider and wider range of problems types as people experiment with its potential. The use of cases as a way of recording situated knowledge will certainly lead to an increase in the use of CBR in computer assisted learning. Moreover, the ability of such systems to continually learn from interactions with new students makes them particularly appealing [Kolodner, 93].

Khris Hammond a pioneer of CBR in the US has proposed dividing CBR research into three categories according to their fundamental goals (note he intends that the categories should be inclusive and not exclusive and should stimulate debate not stifle it):

- *true faith CBR* - relating to the cognitive science and AI theoretical issues of CBR,
- *hard-core CBR* - testing and refining true faith theories in challenging practical applications, and
- *CBR-lite* - using selected CBR techniques, such as case-retrieval, to solve particular problems (e.g., help-desks).

Khris Hammond believes that research into each of these three categories is bona fide CBR research. CBR needs the cognitive science and AI theorists as well as the hard-core researchers and those who apply CBR-lite for commercial profit. As Phil Harmon said "CBR is not rocket science". It is the application of relatively simple and well understood techniques to decision support [Harmon, 92]. CBR is perhaps one of the first fruits of Spring following the AI Winter and as such the AI community has good reason to feel more optimistic about its future.

Acknowledgements

This work was partially funded by EPSRC project number GR/J42496. The authors would like to thank all the software companies that gave evaluation copies of their software and information about their products, and all the CBR researchers throughout the world who found references for us and gave us their comments on aspects of CBR.

Bibliography

This bibliography refers to work cited in this paper. An accompanying paper "A Categorised Bibliography of Case-Based Reasoning" [Marir & Watson, 94] gives a more detailed bibliography of CBR.

Aamodt, A., (1989). Towards robust expert systems that learn from experience - an architectural framework. In, EKAW-89: Third European Knowledge Acquisition for Knowledge-Based Systems Workshop, Boose, J., Gaines, B. & Ganascia J.-G. (eds.), pp.311-326. Paris, July 1989.

Aamodt, A. (1991). *A Knowledge intensive approach to problem solving and sustained learning,* PhD. dissertation , University of Trondheim, Norwegian Institute of Technology, May 1991. University Microfilms PUB 92-08460.

Aamodt, A. & Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(i): pp 39-59.

Acorn, T. & Walden, S. (1992). SMART: Support management cultivated reasoning technology Compaq customer service. In, *Proceedings of AAAI-92* . Cambridge. MA: AAAI Press/MIT Press.

Alexander, J.H., Freiling, M.J., Shulman, S.J., Staley, J.L., Rehfuss, S. & Messick, S.L. (1986). Knowledge Level Engineering: ontological analysis. *AAAI-86*, 2: pp963-68.

Alterman, R. (1986). An adaptive planner. In Proceedings of AAAI-86.: AAAI Press/MIT Press. Cambridge, MA, US

Alterman, R.(1988). Adaptive planning. *Cognitive Science* 12: pp.393-422.

Alterman, R. (1989). Panel discussion on case representation.In, *Proceedings of the Second Workshop on Case-Based Reasoning*, Pensacola Beach. FL, US.

Althoff, K.D. (1989). Knowledge acquisition in the domain of CBC machine centres: the MOLTKE approach. In, *EKAW-89, Third European Workshop on Knowledge-Based Systems*, Boos, J., Gaines, B. & Ganascia, J.G. (eds.), pp.180-95. Paris, July 1989.

Aleven, V. & Ashley, K.D. 1992. Automated generation of examples for a tutonal in case-based argumentation, In *Proceedings, Second Internalional Conference on Intelligent Tutoring Systems* (ITS 92), Montreal, eds. C. Frasson, G.

Gauthier, & G.I McCallan. Springer Verlag. Berlin.

Ashley, K.D. (1988). Arguing by Analogy in Law: A Case-Based Model. In D.H. Helman (Ed.), *Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science, and Philosophy*. D. Reidel.

Bachant, J., & McDermott, J., (1984). R1 Revisited: Four years in the Trenches. *The AI Magazine,* 5(iii).

Bain, W.M. (1986). *Case-Based Reasoning: A Computer Model of Subjective Assessment*. Ph.D. Thesis, Yale University, Yale, CT, US.

Bareiss, E. R.,(1988). *PROTOS: A Unified Approach to Concept Representation, Classification, and learning*. Ph.D. thesis, Department. of Computer Science, University of Texas.

Barletta, R & Mark, W., (1988). Explanation-based indexing of cases. In, *Proceedings of the Seventh National Conference on Artificial Intelligence*. Minneapolis, MN, US.

Barletta, R., (1991). An introduction to case-based reasoning. *AI Expert*, August 1991, pp.42-49.

Bench-Capon, T.J.M. & Coenen, F. (1992). The Maintenance of Legal Knowledge Based Systems. *AI Review*, 6: pp.129-43.

Birnbaum, L. & Collings, G. (1989). Remindings and Engineering Design Themes: A Case Study in Indexing Vocabulary. In, *Proceedings of the Second Workshop on Base-Based Reasoning*, Pensacola Beach, FL.

Boose, J.H. (1989). A survey of knowledge acquisition techniques and tools. *Knowledge Acquisition* 1(i).

Branting, K. (1991). Exploiting the complementarity of rules and precedents with reciprocity and fairness. In, *Proceedings of the Case-Bases Reasoning Workshop 1991, Washington, DC, May 1991.* Sponsored by DARPA. Morgan Kaufmann, pp.39-50.

Brooke, S. & Jackson, C. (1991). Advances in elicitation by exception. In, *Proc. 1st SGES Int. Workshop on Knowledge Based Systems Methodologies*. British Computer Society SGES, pp.70-8.

Burton, A.M., Shabolt, N.R., Rugg, G. & Hedgecock, A.P. (1988). Knowledge elicitation techniques in classification domains. In, *Proceedings of ECAI-88: The 8th European Conference on Artificial Intelligence*.

Chandrasekaran, B. (1986). Generic tasks in knowledge-based reasoning: high level building blocks for expert system design. *IEEE Expert*, 1(iii): pp.23-30.

Chandrasekaran, B. (1990). Design problem solving: a task analysis. *AI Magazine*, Winter 1990: pp.59-73.

Clancey, W.J., (1985). Heurestic Classification. *Artificial Intelligence,* 27: pp289-350.

Coenen, F. & Bench-Capon, T.J.M. (1992). Maintenance and Maintainability in Regulation Based Systems. *ICL Technical Journal*, May 1992, pp.76-84.

Collins, G. (1987). Plan Creation: Using Strategies as Blueprints. Ph.D. Thesis, Department of Computer Science, Yale University, New Haven, CT, US.

Costas, T., & Kashyap (1993). Case-Based Reasoning and Learning in Manufacturing with TOTLEC Planner. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(iv) July/August 1993.

David B.S. (1991). Principles for case representation in a case-based aiding system for lesson planning. In, *Proceedings of the Workshop on Case-Based Reasoning*, Madison Hotel, Washington, 8-10 May, 1991.

Dearden, A.M. & Bridge, D.G. (1993). Choosing a reasoning style for a knowledge-based system: lessons from supporting a

help desk. *The knowledge engineering review*, 8(iii): pp.210-22.

Diaper, D. (1989). *Knowledge Elicitation: Principles, Techniques and Applications*. Ellis Horwood Ltd.

Domeshek, E., (1993). A case study of case indexing: Designing index feature sets to suit task demands and support parallelism. In, *Advances in connectionnist and neural computation theory, Vol.2: Analogical connections*, eds. J. Barenden and K. Holyoak, Norwood, NJ. US.

DTI (1992). *Knowledge-Based Systems Survey of UK Applications*. Department of Trade & Industry, UK.

Falkeneheimer, B., Forbus, K.D. & Gentner, D. (1986). The structure mapping engine. In, *Proceeding of the Sixth National Conference on Artificial Intelligence*, Philadelphia, PA, US.

Farrel, R., (1987). Intelligent case selection and presentation. In, *Proceedings of the tenth International Joint Conference on Artificial Intelligence, IJCAI-87*, 1: pp174-76.

Gallaire, H., Minker, J. & Nicolas, J.M., (1981)**.** *Advances in Database Theory, Vol.1,* Plenum, New York.

Gentner, D. (1983). Structure mapping - a theoretical framework for analogy. *Cognitive Science*, 7: pp.155-70.

Goodman, M. (1989). CBR in battle planning. In *Proceedings of the Second Workshop on Case-Based Reasoning*, Pensacola Beach, FL, US.

Hammond, K.J. (1986). CHEF: A Model of Case-Based Planning. In *Proc. American Association for Artificial Intelligence*, *AAAI-86,* August 1986. Philadelphia, PA, US.

Hammond, K J. (1987). Explaining and Repairing Plans that Fail. In, *Proceedings International Joint Conferences on Artificial Intelligence*, IJCAI-87, August, Milan, Italy.

Hammond, K.J. (1989). On Functionally Motivated Vocabularies: An Apologia. In, *Proceedings of the Second Workshop on Case-Based Reasoning*, Pensacola Beach, FL, US.

Harmon, P. (1992). Case-based reasoning III. *Intelligent Software Strategies*, 8(i).

Hayes-Roth, F., Waterman, D. & Lenat D., eds. (1983). *Building expert systems*. Addison Wesley, Reading, MA, US.

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

Hennessy, D. & Hinkle D., (1992). Applying Case-Based Reasoning to Autoclave Loading. *IEEE Expert*, 7(v): pp.21-6.

Hinrichs, T.R. (1992). *Problem solving in open worlds*. Lawrence Erlbaum Associates.

Kass, A. (1986). Modifying explanations to understand stories. In, *Proc. 8th. Annual Conf. of the Cognitive Science Society*. Lawrence Erlbaum Associates, Hillsdale, NJ, US.

Keane, M., (1988). Where's the Beef? The absence of pragmatic factors in theories of analogy. In, *ECAI-88*: pp.327-32.

Kitano, H. (1993). Challenges for massive parallelism. In, Proc. *13th. Int. Conference on Artificial Intelligence*, IJCAI-93: pp813-34.

Kolodner, J. L. (1983a). Maintaining Organization in a Dynamic Long-Term Memory. *Cognitive Science*, 7(iv): pp.243-80.

Kolodner, J. L. (1983b). Reconstructive Memory: A Computer Model. *Cognitive Science*, 7(iv): pp.281-28.

Kolodner, J. L. (1993). *Case-Based Reasoning*. Morgan Kaufmann.

Koton. P. (1989). *Using experience in learning and problem solving*. Massachusetts Institute of Technology, Laboratory of Computer Science, Ph.D. Thesis MIT/LCS/TR-441.

Leake, D.B. Case-based reasoning. *The knowledge engineering review*, 9(I): pp.61-64.

Lebowitz, M., (1987). Experimental with incremental concept formation: UNIMEM. *Machine Learning*, 2(ii): pp 103-38.

Lopez, B., Plaza, E. (1993). Case-Base Planning for medical diagnosis. *Methodologies for Intelligent Systems, 7th. International Symposium, ISMIS-93*. Lecture Notes in Artificial Intelligence 689, Springer Verlag.

Magaldi, R.V. (1994). CBR for Troubleshooting Aircraft on the Flightline. In, Proc. *IEE Colloquium on Case-Based Reasoning: Prospects for Applications*, Digest No: 1994/057, pp.6/1-6/9.

Maher, M.L. & Zhang, D.M. (1991). CADSYN: using case and decomposition knowledge for design synthesis. In *Artificial Intelligence in Design*, Gero, J.S. (ed.), Butterworth-Heinmann. Oxford. UK.

Marir, F. & Yip Y.J.(1992) An Inference System Based on the Compiled Approach for the Relational Database Systems. *In the ITI Papers of the University of Salford*, :, pp.47-72, May 1992.

Marir, F. (1993). *An Integration approach for the deductive database systems: Enhancing the relational database system with a logic inference based on a compiled approach*. Ph.D. Thesis, University of Salford, UK.

Marir, F., & Watson, I.D. (1994). Case-Based Reasoning: A Categorised Bibliography. *The Knowledge Engineering Review*, Vol. 9 No. 4: pp.355-381.

Milne, R. & Nelson, C. (1994). Knowledge Guided Data Mining. In, Proc. *IEE Colloquium on Case-Based Reasoning: Prospects for Applications*, Digest No: 1994/057, pp.10/1-10/3.

Minker, J.,(1988). Perspectives in Deductive Databases. J*ournal. of Logic Programming,* 5:pp.33-60.

Moore, C.J., Lehane, M.S. & Proce, C.J. (1994). Case-Based Reasoning for Decision Support in Engineering Design. In, Proc. *IEE Colloquium on Case-Based Reasoning: Prospects for Applications*, Digest No: 1994/057, pp.4/1-4/4.

Moorman, K.,& Ram, A. (1992). A Case-based approach to reactive control for autonomous robots. In *Proceedings of the AAAI Fall Symposium on AI for Real-World Autonomous Robots*, AAAI Press/MIT Press, Cambridge, MA, US

Motta, E., Rajan, T. & Eisenstadt, M. (1989). A methodology and tool for knowledge acquisition in KEATS-2. In, *Topics in Expert System Design: Methodologies and Tools*. Guida, G. & Tasso, C. (eds.), pp.297-322. North-Holland.

Navichandra, D. (1991). Exploration and innovation in design:towards a computational model. Springer Verlag, New York. NY, US.

Owens,C., (1993). Integrating feature extraction and memory search. *Machine Learning* 10(iii): pp.311-40.

Oxman, R.E., (1993a). PRECEDENTS: Memory structure in design case libraries. In *CAAD Futures 93*, Elsevier Science Publishers.

Oxman, R.E., (1993b). Case-based design support: Supporting architectural composition through precedent libraries. *Journal of Architectural Planning Research*.

Pearce, M., Ashok K.G., Kolodner, J.L., Zimring, C. & Billignton, R. (1992) Case-Based Support- A case study in Architectural Design.*IEEE Expert* Oct. 1992.

Porter, B.W. & Bareiss, E.R. (1986). PROTOS: An experiment in knowledge acquisition for heuristic classification tasks. *In Proceedings of the First International Meeting on Advnces in Learning (IMAL)*, Les Arcs, France, pp.159-74.

Quinlan, J.R. (1979). Induction over large databases. *Rep. No. HPP-79-14, Heuristic Programming Project*, Computer Science Dept., Stanford University, US.

Ram, A., Arkin, R.C., Moorman, K. & Clark, R.J. (1993). Case-based reactive navigation: A case-based method for on-line selection and adaptation of reactive control parameters in autonomous robotic systems. Georgia Institute of Technology, College of Computing Technical report n0. GIT-CC-92/57, Atlanta, US

Richter, A.M. & Weiss, S. (1991). Similarity, uncertainty and case-based reasoning in PATDEX. In, *Automated reasoning, essays in honour of Woody Bledsoe. Kluwer* R.S. Boyer (ed.): pp249-265.

Reisbeck, C.K, & Schank, R.C. (1989). *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates, Hillsdale, NJ, US.

Rissland, E.L., & Skala , D.B. (1989). Combining case-based and rule-based reasoning: A heurestic approach. In, *Eleventh International Joint Conference on Artificial Intelligence*, IJCAI-89: pp. 524-30, Detroit, Michigan.

Schank, R.C. & Abelson, R.P. (1977). *Scripts, Plans, Goals and Understanding*. Erlbaum, Hillsdale, New Jersey, US.

Schank, R. (1982). *Dynamic memory: a theory of reminding and learning in computers and people*. Cambridge University Press, Cambridge, UK.

Sharma, S. & Sleeman, D. (1988). REFINER: A Case-Based Differential Diagnosis Aide for Knowledge Acquisition and Knowledge Refinement. In, *EWSL 88; Proc. European Working Session on Learning*: pp201-10.

Simoudis, E. (1992). Using Case-Based Retrieval for Customer Technical Support. *IEEE Expert*, 7(v): pp.7-13.

Simoudis, E., Mendall, A. & Miller, P. (1993). Automated support for developing retrieve-and-propose systems. In *Proceedings of Artificial Intelligence XI Conference*, Orlando, Florida.

Simpson, R. L. (1985). *A Computer Model of Case-Based Reasoning in Problem Solving: An Investigation in the Domain of Dispute Mediation*. Technical Report GIT-ICS-85/18, Georgia Institute of Technology, School of Information and Computer Science, Atlanta ,US.

Skalk, D.B. (1992). Representing cases as knowledge sources that apply local similarirty metrics. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*. Lawrence Erlbaum Associates.

Slade. S. (1991): Case-based reasoning: A research paradigm. *AI Magazine* 42-55.

Smith, E.E. Adams, N. & Schorr, D. (1978). Fact retrieval and the paradox of interference. *Cognitive Psychology*, 10: pp.438-64.

Steels, L. (1990). Components of expertise. *AI Magazine*, Summer 1990: pp.28-50.

Sycara, E. P. (1987). *Resolving adversial conflicts: An approach to Integrating Case-Based and Analytic Methods*. Technical Report GIT-ICS-87/26, Georgia Institute of Technology, School of Information and Computer Science, Atlanta GA.

Sycara, K. (1992). CADET: a case-based syntehsis tool for engineering design. *International Journal for Expert Systems*. 4(ii): pp.157-88.

Sycara, K., & Navichandra, D (1992). Retrieval strategies in a case-based design system. In, *Artificial Intelligence in engineering design, Vol.2*, C.Tong and D.Sriram ed.. Academic Press.

Tulving, E. (1977). Episodic and semantic memory. In, *Organisation of Memory* (Tulving, E. & Donaldson, W. Eds.), pp.381-

403. Academic Press.

Vargas, J.E., & Raj, S. (1993). Developing maintainable expert systems using case-based reasoning. *Expert Systems,* 10(iv): pp.219-25.

Venkatamaran, S., Krishnan, R. & Rao, K.K. (1993). A rule-case based system for image analysis. In, Proc. *1st. European Workshop on Case-Based Reasoning, Posters & Presentations*, 2: pp.410-15.

Watson, I.D., Basden, A. & Brandon, P.S. (1992a). The Client Centred Approach: Expert System Development. *Expert Systems* 9(iv): pp.181-88.

Watson, I.D., Basden, A. & Brandon, P.S. (1992b). The Client Centred Approach: Expert System Maintenance. *Expert Systems* 9(iv): pp189-96..

Watson, I.D., & Abdullah, S. (1994). Developing Case-Based Reasoning Systems: A Case Study in Diagnosing Building Defects. In, Proc. *IEE Colloquium on Case-Based Reasoning: Prospects for Applications*, Digest No: 1994/057, pp.1/1-1/3.

Wielinga, B.J., Schreiber, A.Th. & Breuker, J.A. (1992). KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition* 4(i).

Wittgentstein, L. (1953). *Philosophical Investigations*. Blackwell.

Yang, S., & Robertson, D. (1994). A case-based reasoning system for regulatory information. In, Proc. *IEE Colloquium on Case-Based Reasoning: Prospects for Applications*, Digest No: 1994/057, pp.3/1-3/3.

top of page