Is CBR a Technology or a Methodology?

Ian Watson

AI-CBR
University of Salford, Salford, M5 4WT, UK
ian@ai-cbr.org
www.ai-cbr.org

Abstract.

This paper asks whether case-based reasoning is an AI technology like rule-based reasoning, neural networks or genetic algorithms or whether it is better described as a methodology for problem solving, that may use any appropriate technology. By describing four applications of CBR, that variously use: nearest neighbour, induction, fuzzy logic and SQL, the author shows that CBR is a methodology and not a technology.

1 Introduction

Artificial Intelligence (AI) is often described in terms of the various technologies developed over the last three or four decades. Technologies such as logic programming, rule-based reasoning, neural networks, genetic algorithms, fuzzy logic, constraint based programming and others. These technologies are characterised by specific programming languages or environments (e.g., Prolog or rule-based shells) or by specific algorithms and techniques (e.g., the Rete algorithm or back propagation). Each also has, to a lesser or greater extent, laid down particular ways or methods of solving problems (e.g., A*, depth first search, generate and test) that best use the characteristics of each technology.

Case-based reasoning (CBR) is a relative newcomer to AI and is commonly described as a technology like the ones listed above. This paper will show, by examining four very different CBR applications, that CBR describes a methodology for problem solving but does not prescribe a specific technology. The first section of the paper briefly describes CBR and identifies what characterises a methodology in this context. The next four sections each describe an application whose authors each felt could be described as case-based reasoning. The paper then concludes with a summary and discusses the implications of viewing CBR as a methodology.

2 Case-based reasoning

CBR arose out of research into cognitive science, most prominently that of Roger Schank and his students at Yale University [Schank & Abelson, 1977; Schank, 1982; Koldner, 1983 Hammond, 1988]. It is relevant to the argument presented in this paper that CBR's origins were stimulated by a desire to understand how people remember information and are in turn reminded of information; and that subsequently it was recognised that people commonly solve problems by remembering how they solved similar problems in the past. The classic definition of CBR was coined by Reisbeck and Schank [1989]:

"*A case-based reasoner solves new problems by adapting solutions that were used to solve old problems.*"

Note that this definition tells us "what" a case-based reasoner does and not "how" it does what it does. In a little more detail CBR is commonly described by the CBR-cycle.

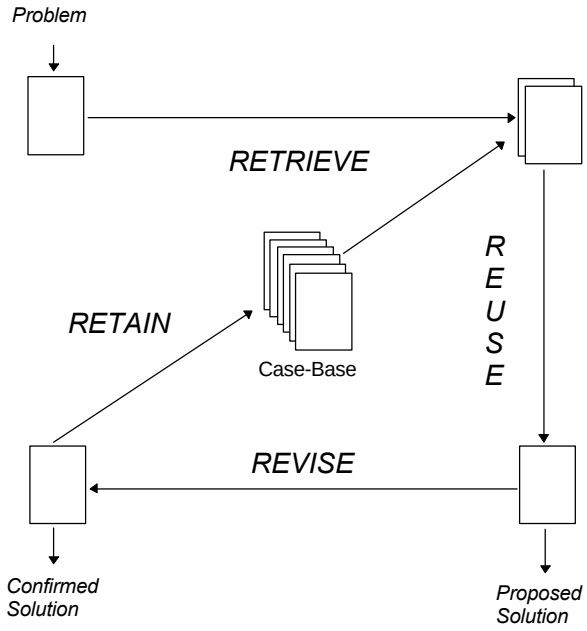

Fig. 1. The CBR-cycle after Aamodt & Plaza, 1994

This cycle comprises four activities (the four-REs):

1. Retrieve similar cases to the problem description
2. Reuse a solution suggested by a similar case
3. Revise or adapt that solution to better fit the new problem if necessary
4. Retain the new solution once it has been confirmed or validated.

Once again, what is being described here is a process or methodology for solving problems and not a specific technology. Peter Checkland describes a methodology as:

> *an organised set of principles which guide action in trying to 'manage' (in the broad sense) real-world problem situations* [Checkland & Scholes, 1990 p.5]

The CBR-cycle fits very nicely into this definition of a methodology as a "*set of principles which guide action*".

What then are the set of principles which guide CBR? The first of these is a desire by the problem solver to solve a problem by explicitly trying to reuse a solution from a similar past problem. Thus, a case-based reasoner must retrieve cases from a case-library and in someway assess the similarity of cases in the library to the current problem description. Second, a CBR system should attempt to reuse the solution suggested by a retrieved case, either with or without revision. Finally, a CBR system should seek to increase its knowledge by retaining new cases.

The subsequent sections will show how four different applications use this set of principles, defined as CBR, to solve real-world problems.

3 CBR using nearest neighbour

Nearest neighbour techniques are perhaps the most widely used technology in CBR since it is provided by the majority of CBR tools [Watson, 1997]. Nearest neighbour algorithms all work in a similar fashion. The similarity of the problem (target) case to a case in the case-library for each case attribute is determined. This measure may be multiplied by a weighting factor. Then the sum of the similarity of all attributes is calculated to provide a measure of the similarity of that case in the library to the target case. This can be represented by the equation:

$$Similarity(T,S) = \sum_{i=1}^{n} f(T_i, S_i) \times w_i$$

where:

T is the target case
S is the source case
n is the number of attributes in each case
i is an individual attribute from 1 to *n*
f is a similarity function for attribute *i* in cases *T* and *S* and
w is the importance weighting of attribute *i*

This calculation is repeated for every case in the case-library to rank cases by similarity to the target. Algorithms similar to this are used by most CBR tools to perform nearest neighbour retrieval. Similarities are usually normalised to fall within a range of zero to one (where zero is totally dissimilar and one is an exact match) or as a percentage similarity where one hundred percent is an exact match. The use of nearest neighbour is well illustrated by the Wayland system [Price & Pegler 1995].

3.1 Wayland - setting up aluminium pressure die-casting machines

Wayland is a CBR system that advises on the set up of aluminium pressure die-casting machines. Wayland was implemented using a very simple CBR called CASPIAN [Pegler & Price, 1996], which can be downloaded from the Internet (www.aber.ac.uk/~cjp/getting-caspian.html). Pressure die casting involves injecting molten metal at very high pressure into a mould (a die), where it cools to make a casting. Machine settings are critical for successful pressure die casting, and there is a compromise between factors such as the cost of producing the casting, maximising the die life, and the quality of the final product. The die parameters are strongly interrelated, making the problem non-decomposable. A change in one parameter can be compensated for by altering another.

CBR is an appropriate technology for this problem, because each foundry will tend to have a particular way of working. Engineers refer to records of previous dies with similar input requirements, and adjust the parameters for a similar die to reflect the different requirements of the new die being built. The records of previous dies are good examples of working compromises between the different operating requirements: such compromises might well have been found by costly adjustments performed in the foundry after the die was built.

```
CASE INSTANCE die_no_5014 IS
        weight_of_casting = 240.00;
        weight_of_casting_and_overflows = 310.00;
        weight_of_total_shot = 520.00;
        no_of_slides = 0.00;
        projected_area_of_casting = 19.50;
        total_projected_area = 35.50;
        average_no_of_impressions = 1.00;
        machine_type = t400;
        metal_type = lm24;
SOLUTION IS
        imagefile = 'dn5014.gif';
        gate_velocity = 6414.09;
        cavity_fill_time = 13.77;
        length_of_stroke = 3.10;
        percentage_fill = 16.24;
        gate_area = 135.00;
        gate_width = 90.00;
        gate_depth = 1.50;
        plunger_velocity = 225.00;
        pressure_on_metal = 8000.00;
        tip_size = 70.00;
        cycle_time = 35.00;
END;
```

Fig. 2. A Case from Wayland

Wayland automates the identification of past dies with similar characteristics, alters the die settings to take into account the differences between the past die and the new one being designed, and validates that the new solution is within design limits.

Wayland has a case base of some 200 previous die designs, extracted from a database of records of actual die performance maintained at the foundry. Only dies with satisfactory performance had their values entered into the case base, so the foundry personnel are confident that each case provides a good basis for calculating new solutions. Cases are fixed format records, with a field for each of the values shown in Figure 2. Some of the fields may be blank, if complete records for a die have not been available. A typical case representation in Wayland is shown in Figure 2.

Cases are retrieved using an algorithm similar to that described above. Each of the retrieved cases is assigned an overall match value by assigning a match score to each field and summing the total. Each field is given a weight which expresses its significance (e.g. the number of impressions is an important field to match: it specifies how many of the parts are made at once in the die). The case with the highest overall mark is the best match. After a case is retrieved it then has adaptation rules applied to it in order to produce the correct machine settings.

Once a case has been accepted, and the die casting has been found to be successful in practice, the case is entered into Wayland's case-base by an engineer, thus, completing the CBR-cycle.

4 CBR using induction

Induction techniques are commonly used in CBR since many of the more powerful commercially available tools provide this facility (e.g., KATE from AcknoSoft, ReCall from ISoft, CBR-Works, from TecInno, and ReMind from Cognitive Systems [Watson, 1997]. Induction algorithms, such as ID3, build decision trees from case histories. The induction algorithms identify patterns amongst cases and partition the cases into clusters. Each cluster contains cases that are similar. A requirement of induction is that one target case feature is defined (i.e., the feature that the algorithm will induce). Essentially the induction algorithms are being used as classifiers to cluster similar cases together. It is assumed (usually correctly) that cases with similar problem descriptions will refer to similar problems and hence similar solutions.

4.1 Troubleshooting CFM 56-3 Engines on Boeing 737s

A good example of the use of inductive techniques for CBR was described by Richard Heider of CFM-international [Heider, 1996]. The project, called Cassiopee, developed a decision support system for the technical maintenance of the CFM 56-3 engines used on Boeing 737 jets. One of the business motivations of this project, in addition to improving problem diagnostics, was to create a corporate memory of troubleshooting knowledge (the *retain* part of the CBR-cycle).

30,000 cases were obtained from a database of engine failure descriptions. Each failure report contained both a structured section that described the failure symptom (e.g., high oil consumption, abnormal noise, thrust deficiency, etc.), and the faulty equipment (i.e., a list of engine parts that needed replacing or maintaining), and a free form text narrative describing the failure event. The textual narratives were analysed by maintenance specialists to identify a further 70 technical parameters that further defined the failure symptoms. Eventually 1500 cases were selected by a specialist as being representative of the range of engine failures. These became Cassiopee's case-base.

The induction algorithm of the tool KATE generated a fault tree from these cases extracting relevant decisions knowledge from the case histories. Retrieval of a similar case is obtained by walking the fault tree to find the cluster of cases that are most similar to the problem description. Once a fault tree is generated retrieval is extremely fast.

In use airline maintenance crews are prompted (via Windows dialogs) to select a failure symptom and to provide additional information about the symptom. The system uses the induced fault tree to find the case or cluster of cases that are most similar to the problem description and provides a list of possible solutions. The cases that provide the solutions can be browsed by the users to help them confirm or reject the solutions.

5. CBR using fuzzy logic

Fuzzy logics are a way of formalising the symbolic processing of fuzzy linguistic terms, such as excellent, good, fair, and poor, which are associated with differences in an attribute describing a feature [Mendel, 1995]. Any number of linguistic terms can be used. Fuzzy logics intrinsically represent notions of similarity, since good is closer to excellent than it is to poor. For CBR A fuzzy preference function can be used to calculate the similarity of a single attribute of a case with the corresponding attribute of the target.

For example, In Figure 3, a difference of 1 unit in the values of an attribute would be considered excellent, a difference of 2 would be good, 3 would be fair, and 4 would be poor. This rating is then transformed into the fuzzy preference function in Figure 3. The result of using fuzzy preference functions is a vector, called the fuzzy preference vector. The vector contains a fuzzy preference value for each attribute. The values in this vector can be combined, through weighted aggregation, to produce a robust similarity value. The use of fuzzy preference functions allows for smooth changes in the result when an attribute is changed unlike the large changes that are possible when step functions are used. A fuzzy preference function is used to transform a quantifiable value for each attribute into a qualitative description of the attribute that can be compared with the qualitative description of other attributes. Thus, A fuzzy preference function allows a comparison of properties that are based on entirely different scales such as cost measured in cents per pound and spectral curve match measured in reflection units.

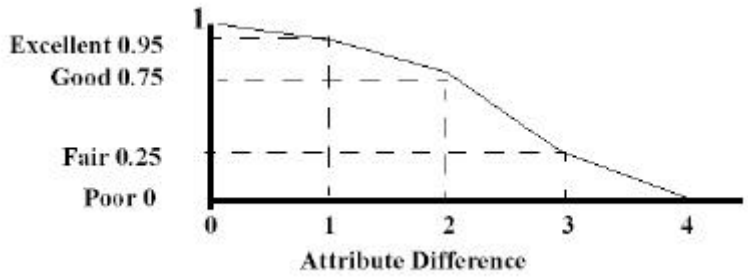

Fig. 3. A Fuzzy Preference Function, after Cheetham & Graf, 1997

5.1 Colour matching plastics at General Electric

A case-based reasoning system for determining what colorants to use for producing a specific colour of plastic was created at GE and has subsequently been patented by them. The selection of colorants needs to take many factors into consideration. A technique that involved fuzzy logic was used to compare the quality of the colour match for each factor. The system has been in use for two years at a growing number of GE Plastics sites and has shown significant cost savings [Cheetham & Graf, 1997].

When presented with a required colour for a new batch of plastic engineers at GE would select the closet match from samples on thousands of colour swatches in a reference collection. The colour formulae of dies from the closest matching swatch would be reused or adapted slightly to produce the required new colour. A swatch of the new colour would then be created and added to the reference collection. This is a pure case-based process being performed by people.

Based on discussions with experts and work to classify previous matches into various sets of linguistic terms GE were able to create fuzzy preference function for each of the following attributes of the colour match:

- colour similarity,
- total colorant load,
- cost of colorant formula,
- optical density of colour, and
- colour shift when moulded under normal and abusive conditions.

Each of the above properties including spectral colour match, loading level, cost, optical density, and colour shift due to processing conditions, is based on different scales of units. But, by mapping each of these properties to a global scale through the use of fuzzy preferences and linguistic terms such as excellent, good, fair, and poor, it was possible to compare one attribute with another. Then these values were input into a typical nearest neighbour algorithm to provide a summed, weighted and normalised score for each colour sample. Thus, fuzzy logic is being used to assess similarity in this system.

6 CBR using databases

At its most simple CBR could be implemented using database technology. Databases are efficient means of storing and retrieving large volumes of data. If problem descriptions could make well formed queries it would be straightforward to retrieve cases with matching descriptions. The problem with using database technology for CBR is that databases retrieve using exact matches to the queries. This is commonly augmented by using wild cards, such as "WEST*" matching on "WESTMINSTER" and "WESTON" or by specifying ranges such as "< 1965". The use of wildcards, Boolean terms and other operators within queries may make a query more general, and thus more likely to retrieve a suitable case, but it is not a measure of similarity.

However, by augmenting a database with explicit knowledge of the relationship between concepts in a problem domain it is possible to use SQL queries and measure similarity.

6.1 SQUAD - sharing experience at NEC

The SQUAD system was developed at NEC in Japan as a software quality control advisory system [Kitano & Shimazu 1996]. Real-world deployment imposed several key constraints on the system. Of these one in particular forced the developers to consider database technology: the system had to be part of the corporate information system and provide a fast response time to over 150,000 users. The use of a commercial RDBMS as a case-manager where each case is represented as a record of a relational database table offered several key advantages such as: data security, data independence, data standardisation and data integrity. The developers of SQUAD were able to create a set of SQL expressions for similarity-based retrieval by referring to abstraction hierarchies. Examples of these are shown in Figure 4.

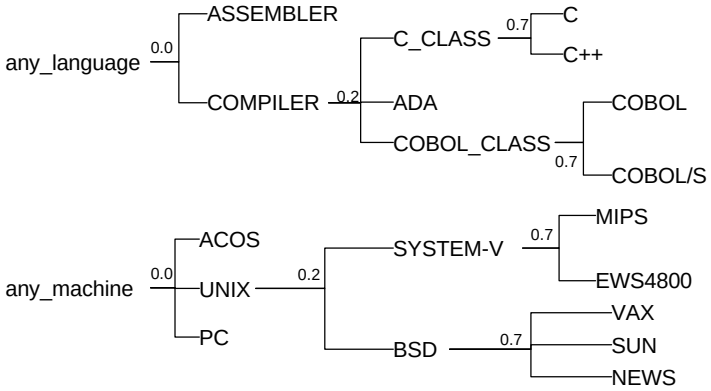

Fig. 4. Examples of Abstraction Hierarchies, after Kitano & Shimazu, 1996

By referring to the abstraction hierarchies for concepts in the problem domain SQUAD can generate a set of similarity values associated with a set of SQL expressions. If a user with a problem identified ADA as the language and VAX as the machine the SQL specifications shown in Table 1 would be generated and sent to the RDBMS as queries. In this way SQUAD is able to assess the similarity of records (cases) returned by the RDMS.

Over 3,000 cases were added to SQUAD each year whilst it was in use resulting in over 25,000 cases, which were accessed by employees all over the global organisation. The developers at NEC believe that this would not have been possible without the scalability, security and robustness provided by a commercial RDBMS system.

Rank	Similarity	SQL Specification (only WHERE clause is shown)
1	1.00	(language = ada) and (machine = vax);
2	0.89	(language = ada) and (machine in (sun, news, …))
3	0.66	(language in (c, c++, cobol, cobol/s)) and (machine = vax);
4	0.54	(language = ada) and (machine in (mips, ews4800, …))
4	0.54	(language in (c, c++, cobol, cobol/s)) and (machine in (sun, news, …));

Table 1. SQL Specifications from SQUAD, after Kitano & Shimazu, 1996

7. Conclusions

Each of the systems described above uses a different technology but they all follow the same set of guiding principles:

- each explicitly attempts to solve problems by reusing solutions to old problems;
- the retrieval of past problems (cases) involves assessing the similarity of the problem to cases in a case-library, and
- once a new problem is solved it is added to the case library to retain the problem solving experience for future reuse.

The developers of the systems described above were therefore correct to describe their systems as case-based reasoners since they adhere to the CBR methodology. If you now accept that CBR is a methodology for problem solving and not a technology you may now be able to see ways of applying it using techniques other than those described here. However, if you now think that CBR can use nearest neighbour, induction, fuzzy logic and database technology you have missed the point of this paper. A case-based reasoner can use any technology provided the system follows the set of principles outlined here.

I believe that viewing CBR as a methodology is important to its continued development. If CBR is viewed as a technology it might seem that research into CBR was largely completed since, for example, nearest neighbour and inductive retrieval are mature and reliable techniques. But if CBR is viewed as a methodology researchers have the challenge of applying any number of technologies. For example, it has been

proposed that neural networks can be used to assess similarity since a NN can tell us, with a degree of certainty, whether two patterns are similar [Thrift, 1989]. Moreover, AI will surely develop new technologies in the future, some of which may prove very suitable for the CBR methodology. Consequently, it is as a methodology that CBR's future is ensured.

8 References

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

Checkland, P. & Scholes, J. (1990). Soft Systems Methodology in Action. Wiley.

Cheetham, W. & Graf, J. (1997). Case-Based Reasoning in Colour Matching. In Proc. *ICCBR-97*, Leake, D. & Plaza, E. (Eds.) LNAI, Springer.

Hammond, K.J. (1988). Case-Based Planning: Viewing planning as a memory task. In, Proceedings of the *DARPA Case-Based Reasoning Workshop*, Kolodner, J.L. (Ed.), Morgan Kaufmann, Calif., US.

Heider, R. (1996). Troubleshooting CFM 56-3 Engines for the Boeing 737 Using CBR & Data-Mining. In, *Advances in Case-Based Reasoning*, Smith, I. & Faltings, B. (Eds.), pp.513-18. Lecture Notes in AI 1168, Springer.

Kitano, H., & Shimazu, H. (1996). The Experience Sharing Architecture: A Case Study in Corporate-Wide Case-Based Software Quality Control. In, *Case-Based Reasoning: Experiences, Lessons, & Future Directions*, Leake, D.B. (Ed.). AAAI Press / The MIT Press, Menlo Park, Calif., US.

Kolodner. J.L. (1983). Reconstructive memory, a computer model. *Cognitive Science*, 7(2), pp.281-328.

Mendel, J. (1995). Fuzzy Logic Systems for Engineering: A Tutorial, In, Proc. of the *IEEE*, 83(3).

Pegler, I., & Price, C.J. (1996) Caspian: A freeware case-based reasoning shell. In, Proceedings of the *2nd UK Workshop on Case-Based Reasoning*. Watson, I. (Ed.), Salford University, Salford, UK.

Price, C.J., & Pegler, I. (1995). Deciding Parameter Values with Case-Based Reasoning. In, *Progress In Case-Based Reasoning*, Watson, I. (Ed.). Lecture Notes in Artificial Intelligence 1020, Springer-Verlag.

Riesbeck, C.K., and Schank, R. (1989). *Inside Case-Based Reasoning*. Northvale, NJ: Erlbaum.

Schank, R., & Abelson, R. (Eds.) (1977). *Scripts, Plans, Goals and Understanding*. Hillsdale, NJ: Erlbaum.

Schank, R. (Ed.) (1982). *Dynamic Memory: A Theory of Learning in Computers and People*. New York: Cambridge University Press.

Thrift, P. (1989). A Neural Network Model for Case-Based Reasoning. In, Proceedings of the *DARPA Case-Based Reasoning Workshop*, Hammond, K.J. (Ed.), Morgan Kaufmann, Calif., US.

Watson, I. (1977). *Applying Case-Based Reasoning: techniques for enterprise systems*. Morgan Kaufmann, Calif., US.