Watson, I. (1994). The Case for Case-Based Reasoning. In, Proc. Information Technology Awareness in Engineering Conference, 21-22 November 1994, London.

The Case for Case-Based Reasoning in Engineering Decision Support

Dr. Ian Watson
AI-CBR, University of Salford
Bridgewater Building
SALFORD, M5 4WT
www.ai-cbr.org
ian@ai-cbr.org

Introduction

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 many of them in manufacturing industries [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. However, despite the undoubted success of knowledge-based decision support systems 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 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

A not untypical story of implementing and deploying KBS was told by Richard Perkins of British Coal's IT division [Perkins, 92]. They found that the cost of developing large complex engineering decision support system was so great that they were not having any significant impact on the business. This was despite the fact that the individual systems were judged a success. (British Coal has recently disbanded its IT division and out-sourced its IT requirements).

Over the last few years a reasoning paradigm and computational problem solving method that seems to address the problems identified above has increasingly attracted more and more attention. Case-based reasoning (CBR) solves new problems by adapting previously successful solutions to similar problems.

- 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 making maintenance easier.

This paper therefore has two objectives: first to outline the techniques and application of CBR to a new audience, and secondly, to demonstrate that the benefits of CBR can be commercially realised. A full review of CBR is available in Watson & Marir [94] and Marir & Watson [94].

What is case-based reasoning?

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

At its most simple case-based reasoning is based on the observation that when we solve a problem we often base our solution on one that worked for a similar problem in the past. An example would be driving to work. When you get in the car in the morning you don't explicitly plan your route, you take the route you usually take. If you meet a traffic jam you may remember how you avoided a similar jam in the past. If you take an alternate route to avoid a jam and it's a success, you will remember it and perhaps use it again in similar circumstances in the future.

CBR is thus a deceptively simple problem solving paradigm that involves matching your current problem against problems that you have solved successfully in the past. The process can be augmented by adapting solutions so they more closely match your current problem.

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*:

1. RETRIEVE the most similar case(s);
2. REUSE the case(s) to attempt to solve the problem;
3. REVISE the proposed solution if necessary, and
4. RETAIN the new solution as a part of a new case.

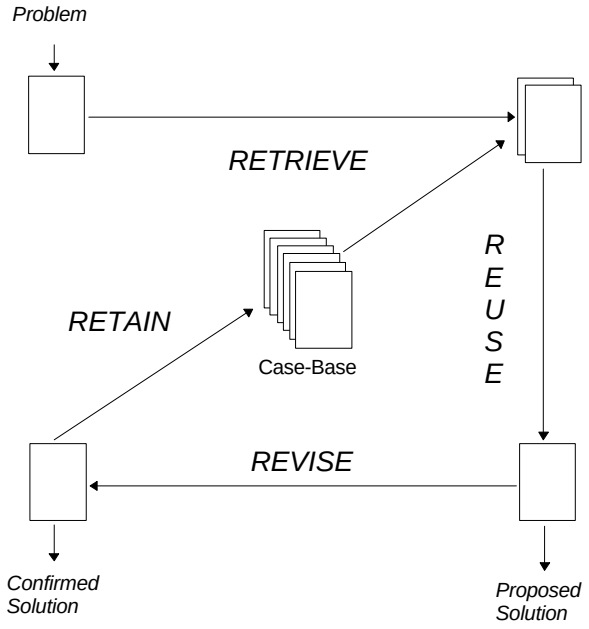

Figure 1 The CBR Cycle [adapted from Aamodt & Plaza, 1994]

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 a weakness of CBR that it encourages human collaboration in decision support.

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 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, 89]. 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.

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. 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.

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 f_i^I and f_i^R 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.
- *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. Sometimes referred to a *reinstantiation* this technique can only be used for cases that are well understood.

CBR applications

This section describes two commercially fielded CBR applications and discusses why CBR contributed to the success of the systems. Both applications are from US defence companies a reflection that the Pentagon, through the DARPA program, is largely responsible for the research and development of CBR.

Lockheed

The first commercially fielded CBR application was by Lockheed in 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 parts 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 model-based 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.

General Dynamics

In their Electric Boat Division the US company General Dynamics builds warships. They had a problem of how to select appropriate mechanical equipment (e.g., valves, heat exchangers,

etc.) during ship design. Most of the problems were *standard* (i.e., they repeated regularly from ship to ship). However, *non-conforming* problems took a considerable amount of time to resolve. The company was particularly frustrated because the non-conforming problems repeated occasionally. They realised that they were wasting time and scarce expertise repeating decisions that had already been made because they had no facility to manage their knowledge.

General Dynamics' problem seemed like a classic knowledge-based system problem.

- Expertise was scarce,
- solutions to problems were known, and
- methods for solving problems were known.

On advice they implemented a rule-based system for the selection and adjustment of valves for on-board pipeline systems. The system was deployed in the late 1980's. The rule-based system worked but was *brittle*. That is, it could solve the standard problems well, but was not reliable at solving the non-conformers. Every time a non-conforming problem was encountered an expert would solve the problem, and knowledge engineers would subsequently have to elicit new rules from the experts (taking up their valuable time) and add them to the KBS, possibly having to modify existing rules and then validate the system and release an update.

The company found that this continual maintenance was unsupportable, and in 1991 a CBR version of the same system was developed. General Dynamics noted several findings:

1. The CBR system was developed in less than half the time of the rule-based system, however, this could partially be explained by the knowledge elicitation that had already taken place to develop the rule-based system.
2. The CBR system was less brittle since new problems frequently were solved by adapting old solutions. These new solutions, once validated, could be added as new cases and thus the system's performance was constantly improving.
3. Consequently maintaining the CBR system was not a problem.

In its first year of deployment the CBR system handled 20,000 non-conformities and made an estimated saving of $240,000 more than recouping the development costs. The system is now being extended to cover a wider range of mechanical devices.

The case for case-based reasoning

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

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, 85] 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, there are five major problems with this approach:

1. knowledge elicitation is difficult
2. knowledge-based decision support systems can be very complex and can take many man years to develop,
3. such systems are frequently slow or require expensive specialised hardware,
4. such systems are often poor at managing large volumes of information, and

5. 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., 83]. The second problem is familiar to any KBS developer and has partially been responsible for the increasing interest in the last few years 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 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, 93]. 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 model-based reasoning:

- 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 and by General Dynamics.
- 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. Indeed, in CBR a system is never *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 - this is important when a decision is required quickly.

- 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.
- CBR systems can *learn* by acquiring new cases making maintenance easier as demonstrated by CLAVIER and General Dynamics
- 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 model-based 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, 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 model-based systems can be very effective and are a relatively mature and well understood technology.

Conclusion

Although CBR as a decision support technology is only about ten years old it has already delivered commercially successful engineering decision support systems. As a comparison neural computing was first proposed in the 1950's and has only recently delivered commercial applications. CBR seems to offer solutions to many of the problems that have beset knowledge engineering since the discipline was founded. Namely, the difficulty of eliciting reliable knowledge, encoding that knowledge and subsequently maintaining systems as knowledge changes. At recent meetings the author has attended many companies using CBR have made the following observations:

- They view CBR as a *low risk* technology as opposed to more traditional AI techniques such as rule-based systems, neural computing or exotica such as fuzzy logics and genetic algorithms - in effect it is easy to sell CBR to senior management.
- They report that their experts are more comfortable recounting cases rather than attempting to distil rules that explain occurrences or behaviour.
- In many instances experts and even end users *author* cases themselves thus giving them a sense of ownership of the system and assisting the system's acceptance. That is the system is not viewed as being developed by boffins who speak predicate calculus and do not really know what the users' problems are.
- They welcome the ability that CBR systems have of acting as knowledge repositories that can easily grow and can be used as an archive of best or even worst practise, thus preventing people from reinventing wheels and avoiding repeating mistakes.

Thus, in conclusion if you are considering implementing an engineering decision support system you would be well advised to at least consider CBR. Mature reliable software tools are

available, and the paradigm overcomes (or avoids) many of the problems often associated with knowledge-based systems.

References

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

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

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

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

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.

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.

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.

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

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

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.

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

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

Perkins, R. (1992). Diagnostic System Designer. *Manufacturing Intelligence*, No. 10 pp.16-19.

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

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

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.

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

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

Watson, I. & Marir, F. (1994) Case-Based Reasoning: A Review. *The Knowledge Enginering Review*. Vol 9. No.4 (in press).

Further information

CBR software tools

The following papers provide reviews of CBR software tools:

Harmon, P. (1992). Case-based reasoning Ill. *Intelligent Software Strategies, 8(1)*.

Watson, I. & Marir, F. (1994) Case-Based Reasoning: A Review. *The Knowledge Enginering Review*. Vol 9. No.4 (in press).

At the time of going to press (Summer 1994) the following software tools with a CBR component are commercially available and supported.

AknoSoft, ***KATE***
58a, Rue du Dessous des Berger
75013 Paris, France
Tel: (33-1) 44 24 88 00
Fax:(33-1) 44 24 88 66

Esteem Software Inc,. ***ESTEEM***
302E. Main street Cambridge City,
IN 47327, USA
Tel: (317) 478-3955
Fax:(317) 478-35550

Inference Corporation, ***ART*Enterprise****,*
CBR Express *and* ***CasePoin****t.*
Inference Europe Ltd
31-37 Windsor Road
SLOUGH
SL1 2ED
Tel: 0753-811855
Fax: 0753-811860

The Haley Enterprise Inc, ***Eclipse***
413 Orchard Street
Sewickley, PA USA 15143
Tel: (412) 741-6420
Fax: (412) 741-6457

Cognitive System Inc., ***ReMind***
220-230 Commercial Street, Boston, MA 02 109,
USA.
Tel: (617) 742-7227
Fax: (617) 742-1139.

Inductive Solution Inc., ***CasePower***
380 Rector Place, Suit 4A,
New York, NY 10280, USA
Tel: (212) 945-0630
Fax:(212) 945-0367

Isoft, ***ReCall***
Chemin de Moulon
F-91190 Gif sur Yvette
France
Tel: (33-1) 69 41 27 77
Fax: (33-1) 69 41 25 32

Internet CBR Sources

The following section presents sources that provide information via the Internet on CBR.

AI-CBR

This is the Internet site covering all aspects of CBR research and practice. Membership is free and members include academics, industrialists, and many of the CBR software vendors. In addition to an electronic conference AI-CBR contains papers and articles on CBR that may be downloaded along with a bibliography of CBR research.

http://www.ai-cbr.org

The European CBR Newsletter

The case-based reasoning electronic newsletter is delivered to the members of the German AK-CBR and to the participants of the EWCBR-workshops Thus, the CBR Newsletter addresses mainly a continental European readership. Its objective is to support an exchange of information, news, and opinions on CBR that relate to both scientific and application-oriented issues. People who want to receive the CBR Newsletter should contact:: Dietmar Janetzko at: dietmar@cognition.iig.uni-freiburg.

Acknowledgements

This work was partially funded by EPSRC project number GR/J42496.