A Case-Based Reasoning Solution to the Problem of Redundant Resolutions of Nonconformances in Large-Scale Manufacturing

Stuart J. Brown and Lundy M. Lewis

By *large-scale manufacturing*, we refer to the manufacture of a few large, complex items over a long period of time as opposed to the manufacture of many smaller items during a shorter period of time. For example, more than five years are required to build one ship, whereas thousands of automobiles are built at a single plant during one year. Further, thousands of person-years go into the making of a single ship. As a result of the complexity of large items and the effort and time required to build them, it is inevitable that episodes of atypical problem solving are unknowingly duplicated. This situation results in lost opportunities and less than optimal production costs.

The Problem

We illustrate this situation with a common engineering problem in

ship building. A particular type of valve is used in some thousand fluid systems on a specific class of ships, and often, it is used in many places on a single fluid system. The operations performed on each valve are "receive," "inspect," "install," "test," and "accept." Each of these operations is controlled by different cognizant engineers, and a different set of cognizant engineers is responsible for each fluid system. When an operation fails, a well-defined investigatory procedure follows. If the investigatory procedure is sufficiently robust, a repair method for the failure is invoked. However, if a repair method is not evident, the investigatory procedure becomes unstructured. Usually, the cognizant engineer confers with colleagues or superiors. When a repair method is found, it is documented but is not made public. An engineer with a similar problem in the future is lucky if someone in his(her) group remembers the old repair method and retrieves the applicable document. More often than not, engineers confront atypical problems afresh, without opportunities to exploit similar past solutions.

In the ship-building industry, these sorts of problems are called nonconformances. Specifically, a *nonconformance* is the failure of a system or subsystem whose repair method is not dictated by standard resolution procedures. Nonconformances occur with all components of mechanical, electric, and structural systems. In our facility, some 35,000 noncomformances occur annually. The problem of redundant resolutions of a nonconformance (RRN) occurs when a nonconformance is researched and resolved in relative isolation. The RRN problem arises because useful expertise that can be applied to nonconformance problems is transient and distributed over both time and space. For example, (1) a particular nonconformance is addressed by different people as a result of personnel turnover, (2) similar nonconformances recur at different times during the construction cycle of systems or subsystems, (3) similar subproblems of large complex nonconformances are addressed on different occasions by different personnel, and (4) a particular nonconformance is addressed by different departments or different people within the same department.

The RRN problem is compounded in large-scale manufacturing because communication among the experts is sparse. The obvious solution to the problem is to identify and categorize nonconformances and their respective problem-solving strategies and resolutions and make this knowledge readily available to all engineers for application to current nonconformances. However, this solution is difficult because knowledge must be extracted from multiple experts, the knowledge must uniformly be represented, and relevant knowledge must be accessible to any one expert.

The Solution

In January 1986, a manufacturing and production engineering project was initiated to demonstrate the feasibility of a paperless nonconformance system using AI technology. The initial approach was to express the requisite knowledge as rules in a production system. A prototype system was developed in KEE on an AI workstation and then later reimplemented in Goldworks on a personal computer. The initial results were promising. However, as the system became increasingly complex, it began to suffer the problems of brittleness and knowledge-acquisition bottleneck. The system could not resolve nonconformances that were not already explicit in the knowledge base, and the experts found it increasingly difficult to express their knowledge as a collection of rules.

The results of an internal research and development project during 1988–1989 demonstrated that case-based knowledge representation is a close match to the properties of the RRN problem (Lewis 1989). By 1990, a case-based reasoning application to the RRN problem was constructed, and by the second quarter of 1990, it was deployed with a sufficiently large, albeit incomplete, knowledge base.

The basic idea of case-based reasoning is to recall, adapt, and execute traces of former experiences in an attempt to deal with a current experience. Former experiences are represented as a library of cases that reside in memory, where individual cases are related through various link types, including abstraction, exemplar, index, and failure links. When confronted with a new problem, a case-based reasoning system retrieves a maximally similar case, and information from the case is adapted to the new problem in an attempt to solve it. If the solution is successful, the new case is embedded in memory for future problem solving. If the solution is unsuccessful, the case is tagged and embedded in memory with the reasons why it did not work.

The characteristics of case-based reasoning that render it suitable for the RRN problem are (1) *case structure*, through which previous paper reports of nonconformances and their solution strategies and resolutions are representable; (2) *similarity metrics*, by which relevant representations of nonconformance reports and solutions are retrieved; (3) *adaptation techniques*, by which the resolution of a current nonconformance problem is derived from a retrieved report; and (4) the *organization of the case library*, through which experience with a current disposition is embedded in memory for future use.

The primary advantages of case-based over rule-based knowledge representation are that the problems of brittleness and knowledge-acquisition bottleneck are less severe. Case-based reasoning systems are designed to adapt an old solution to a new, similar problem. Further,

the expert is more comfortable relating his(her) expertise constrained by case-oriented knowledge representation as opposed to rule-oriented knowledge representation, and the knowledge base is refined and updated during use.

In the following discussion, we provide the groundwork for the development of the system and then describe the system and its benefits. Other problem areas to which case-based reasoning techniques have been applied and from which we borrowed some fundamental ideas include legal domains (Ashley and Rissland 1988; Bain 1986; Rissland and Skalak 1989b), medical domains (Koton 1988), engineering design (Daube and Hayes-Roth 1989; Huhns and Acosta 1988), and software diagnostics (Simoudis and Miller 1990). Overviews of case-based reasoning are provided in Riesbeck and Schank (1989) and Slade (1991). Experimental case-based reasoning systems are discussed in Hammond (1986) and Kolodner, Simpson, and Sycara-Cyranski (1985). A hybrid system consisting of a case-based reasoning component and a qualitative reasoning component is discussed in Koton (1988). A hybrid system consisting of a case-based reasoning component and a rule-based component is discussed in Rissland and Skalak (1989a, 1989b). The production of a generic case-based reasoning tool (CABARET) is in progress at the University of Massachusetts for the purpose of studying issues concerning case-based reasoning (Rissland and Skalak 1989a), and a case-based reasoning shell sponsored by the Defense Advanced Research Projects Agency is discussed in Stottler, Henke, and King (1989).

Groundwork 1: Similarity Metrics for Case-Based Reasoning Systems

Assume that an object can be represented as a list of *n* attribute-value pairs, where each value is numeric. An object is represented as a point in *n*-dimensional space, where the dimensions represent the attributes, and the object is the point whose coordinates are determined by the attribute values. The geometric definition of distance quantifies the similarity between two objects. Smaller distances indicate increasing similarity. A library of objects is represented as points plotted in *n*-dimensions, where clusters of similar objects and unique objects are identifiable (Stottler et al. 1989). To illustrate, suppose the object is a part, and the dimensions of the part are height and weight. The two parts

[height(6),weight(100)], [height(5),weight(110)]

are more similar than

[height(5),weight(100)], [height(2),weight(30)] ,

with the closeness measure of each pair calculated as approximately 10 and 70, respectively.

Assume now that attribute values are symbolic. A first-order definition of similarity is prescribed by the larger number of exact matches of attribute-value pairs of any two objects; that is, a larger number of matches indicates increasing similarity. Although this definition appears to be reasonable, numerous counterexamples can be construed for which high similarity does not warrant the extension of a known attribute of an object to a target object. For example, if two nonconforming parts are determined highly similar based on irrelevant attributes, the fact that the one nonconformance is resolved by method *M* does not warrant the inference that the other one can be resolved by *M*. If it is known, however, the extent to which certain attributes are relevant to a nonconformance, and the values of the relevant attributes match in each case, then the inference of *M* to the other part is more reasonable. Thus, similarity is defined as the larger number of matches of attribute-value pairs relevant to a particular nonconformance. This idea is worked out conceptually in the *theory of determinations* (Davies and Russell 1987). The theory suggests that a library of prior nonconformances and their solutions be augmented with a set of determination rules that record relevance information among sets of attributes and possible nonconformance solutions.

A related approach argues that similarity is a function of the purpose of object comparisons (Kedar-Cabelli 1986). Knowledge about the purpose of the comparison of two objects focuses attention on the relevant attributes of the objects. In this approach, similarity is defined as the number of matches of attribute-value pairs constrained by the purpose of comparison. The approach suggests that a library of nonconformances be examined with an explicit purpose, where determination rules record relevance information between purposes and sets of attributes. To expand the previous example, suppose the purpose of reviewing a library of nonconformances is to collect those nonconformances that involve a certain vendor. A determination rule will have associated the vendor with a range of attributes and values, and thus, a cluster of nonconformance reports are retrieved accordingly.

We note that determination rules preserve semantic information about the relations among sets of attributes. A subsidiary problem is, of course, how determination rules come to exist. In our domain, determination rules are articulated by expert engineers in ship building and are not always accurate. An alternative domain-independent approach, the *structure-mapping theory* (SMT), avoids this problem by couching similarity as a function of syntax only (Gentner 1983). First, we distinguish between attributes, relations, and higher-order predicates, for exam-

ple, $a(X)$, $a(X,Y)$, and $a(b(X,Y),c(Y,Z))$, respectively. The *systematicity principle* states that "a predicate that belongs to a mappable system of mutually interconnecting relationships is more likely to be imported into the target than is an isolated predicate" (Gentner 1983, p. 163). *Similarity*, then, is defined as the degree of match between higher-order predicates whose arguments denote increasingly interconnected relationships. For example, the predicate

[material(diaphragm,*X*), temp(*X*,Degree,Beginning,End)] —> [fail(diaphragm, End + *N*)] ,

where $N = f(X,\text{Degree,Beginning,End})$ and —> denotes causation, states that the failure of a diaphragm is caused by, and is a function of, the material that the diaphragm is made of, the extent to which it is heated, and the interval during which it is heated. SMT suggests that a predicate of this form affords more similarity import than, say, material(diaphragm, *X*). Further examples and empirical support of SMT are provided in Gentner (1983). Although we have not used this technique in our application, we consider it a promising approach to a domain-independent formalization of similarity.

Groundwork 2: Adaptation Techniques for Case-Based Reasoning Systems

A simple adaptation technique of a retrieved case is *structural adaptation*, in which adaptation rules apply directly to a solution stored in a case. Here, we describe four kinds of structural adaptation (Riesbeck and Schank 1989).

First is *null adaptation*. The simplest technique, it directly transfers an old solution to a new problem. Note that this technique can be achieved with methods simpler than case-based reasoning, including lookup tables and associative connectionist systems.

Second is *parameterized adaptation*, which adjusts the solution variables of a target case relative to the solution variables of a source case. For example, a problem variable *X* in a source case can be related to a solution variable *Y* according to the rule "As *X* increases, *Y* decreases." The value of *X* in a target case is compared to the value of *X* in the source case, and the value of *Y* in the target case is instantiated relative to the value of *Y* in the source case. Refer to Bain (1986) and Rissland and Ashley (1986) for further discussion and application of this technique.

Third is the *abstraction-respecialization* technique, which presupposes the organization of attributes of a case library into an is-a hierarchy and considers constraints on possible solutions to a problem in a target

case. The idea is that if a solution of a source case does not satisfy the constraints of the target case, then one should abstract and respecialize over the source attribute and check the new value against the constraints. For example, suppose repair(pump(diaphragm)) is proposed as a solution to faulty(pump) through a source case, and a constraint imposed on the solution of the target case is not(repair(pump(diaphragm))). Abstraction and respecialization over the attribute *faulty(pump)* in the source case might issue replace(pump(diaphragm)), which is consistent with the constraint. The idea of tweaking a solution involves higher abstractions and respecializations over several parts of the source case and working each part back to satisfy multiple constraints of the target case. Refer to Alterman (1986) for further discussion and application of this technique.

Fourth is *critic-based adaptation*, which occurs when a critic repairs a retrieved solution to fit the target case. The repair method is then attached to the solution and embedded in memory for future use. An example of this technique is to reorder the steps in the retrieved solution. For example, suppose a solution to paint(chair) and paint(ceiling) is paint(chair), climb(ladder), paint(ceiling), and descend(ladder). A similar problem, paint(ladder) and paint(ceiling), might propose paint(ladder), climb(ladder), paint(ceiling), and descend(ladder), which the critic will note is unacceptable. One way to repair the solution is to reorder the steps into climb(ladder), paint(ceiling), descend(ladder), paint(ladder). Refer to Hammond (1986) for further discussion and application of this technique.

A more complex adaptation technique is *derivational adaptation*, in which a known method for solving an old problem is tried on a similar new problem (Riesbeck and Schank 1989). For example, suppose a case represents a problem space, including initial and goal states and a method for solving the problem. Let the relevant attributes for measuring similarity between two cases be the initial states and goal states. Suppose the source case is a solution to a particular sliding block puzzle that was solved by representing the problem as A^* search. The case will contain an initial state, a goal state, a set of operators for traversing the problem space, a heuristic function $f^* = g^* + h^*$ that estimates the merit of each state generated, and a sequence of moves representing the solution. One way to adapt the case for solving a similar problem is to rerun the A^* algorithm against the new problem parameters of a similar case. A more complex version of this technique is discussed in Carbonell (1983), in which the relevant dimensions for measuring similarity are the initial state, the goal state, path constraints, and operators. A retrieved case is adapted by reducing the differences between it and a new similar case along each dimension, in concert.

Description and Operation of the System

The current system consists of five modules: (1) an *input module* for the acquisition of resolved nonconformance reports and known information about current unresolved nonconformances, (2) a *reasoning module* that retrieves a select group of (possibly adapted) nonconformance reports, (3) a *display module* that reports potential solutions found by the reasoning module, (4) an *edit module* that allows the user to manually adapt the retrieved solution, and (5) an *accept module* that prints the solution and updates the case library (Brown 1990). Figure 1 shows the system architecture. We discuss the operation of the system in the following subsections. The system was developed on an IBM personal computer using Borland TURBO C and C++.

Input

The input module consists of a *case-acquisiton submodule* that is used to build and edit the case library and a *problem-acquisition submodule* that is used to submit known problem parameters of a current nonconformance. The structure of a case is the same in both submodules. A primary concern in determining case structure is the following: If case structure is overly specific, the case library becomes overly large, and the retrieval and adaptation of a case is similar to a lookup table. However, if case structure is underspecified, the case library is smaller, but complex adaptation techniques are required, and the system approaches analogical reasoning. The structure of a case in our system naturally models previous paper reports of nonconformances and their resolutions. Case attributes include both numeric and symbolic values, and the attributes are mapped directly from existing nonconformance documents. Although the earlier rule-based systems were inadequate, much of the domain knowledge collected for these systems was transferable to the case-based reasoning system. This approach expedited the construction of the case library and resulted in a fair trade-off between adjustment complexity and the number of cases.

Reason

The reasoning module consists of the *retrieve* and *adapt* submodules. The retrieval of previous similar nonconformances is guided by a set of determination rules that represent relevance information between particular nonconformance problems and sets of attributes. The purpose of the determination rules is to focus attention on previous nonconformances and resolutions that are most similar to the current problem. The determination rules are initially provided by experts. For example, an engineer often associates the problem "leaky valve of type *X*" with

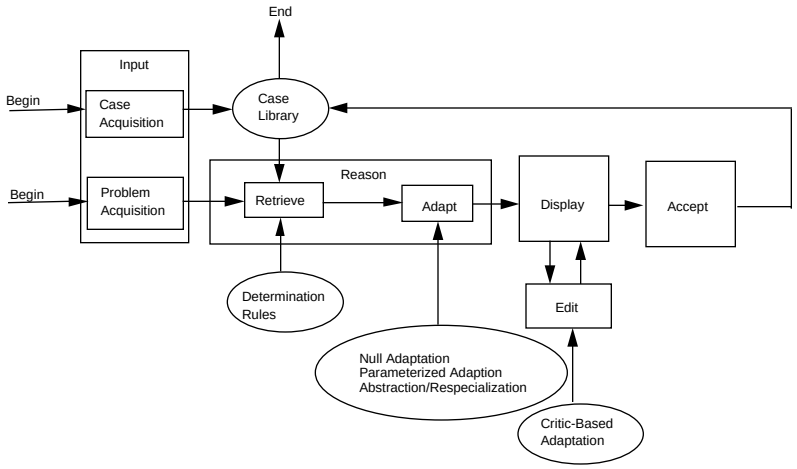

Figure 1. The System Architecture.

the manufacturer of the valve, and this additional piece of knowledge serves to narrow the selection of possibly applicable nonconformances.

Standard nonconformance solutions are stored in a secondary knowledge base, and determination rules can point to this knowledge rather than retrieve a similar, adaptable solution from the case library. This knowledge is the same as the usual investigatory procedures invoked for typical nonconformances.

Structural adaptation techniques are used to adapt the solution of a retrieved case to a target case. Three of the four adaptation techniques are automatically performed by the reasoning module: null adaptation, parameterized adaptation, and abstraction-respecialization.

Null adaptation occurs when determination rules point to a standard nonconformance solution in the secondary knowledge base.

Parameterized adaptation occurs when the system notices that a nonconformance is similar to a prior case in the case library whose solution variables vary proportionally with the variables in its problem definition. A new solution is proposed by adjusting the parameters of the old solution to comply with the requirements of the new problem. For example, some cases contain a variable *X* in the problem definition that represents the difference between the actual and desired amounts of pressure released through a valve, a variable *Y* in the solution that represents the number of quarter turns of an adjustment screw, and an equation relating *X* and *Y*. A similar valve with a similar problem, for which such an equation is unavailable, will use the equation in the source case to propose a solution to the new problem.

The abstraction-respecialization adaptation technique occurs over

those source cases that contain alternative solutions to a problem. Some features of a problem impose constraints on admissible solutions. If a proposed solution is inconsistent with any constraint of the new problem, an alternative solution is proposed. A typical example is the failure of a valve for which the preferable solutions are to repair the valve (most preferred), replace the valve, or consult the manufacturer (least preferred). If during problem acquisition, the user indicated that the valve cannot be repaired, the second solution is proposed instead of the first.

Derivational adaptation techniques, in which a problem-solving strategy is applied to problem parameters of a current nonconformance, are under investigation but were not implemented in the system. Whereas structural adaptation techniques are simple and easy to understand by users, derivational techniques are relatively complex. It is suspected that the derivational technique will discourage users. For similar reasons, we did not use SMT as a similarity metric for retrieving cases.

Display

The display module redisplays the problem and offers possible solutions found by the reasoning module. The user examines the solutions and chooses to either edit a solution or accept one of the solutions offered.

Edit

Critic-based adaptation is used in the edit module when the user knows that the proposed solution is unacceptable and can modify the solution to make it fit the target case. For example, a particular manufacturer's valve cannot be repaired but must be replaced. In the source case, the manufacturer's name was not included in the description of the problem. Everything else being equal, the proposed solution was to repair the valve, which the user knew would not work. Thus, s/he changed the solution to "replace valve" and entered the manufacturer's name in the appropriate slot of the problem description. When the knowledge base is updated with this new information, the reasons why the original solution did not work become implicit in the new case.

Accept

Continuing the previous example, the accept module prints a nonconformance document and updates the case library with the new knowledge. The current method of knowledge update is to add the newly adapted case to the library. Ideally, the original case and the adapted case would be fused into a more general case that covers both problems. This subject is one for future research and development.

Deployment and Discussion of Benefits

In total, 10 person-years went into the development of the current system, some 5 of which were spent on the manufacturing and production engineering project developing the earlier rule-based systems, and 2 person-years were spent on the AI internal research and development project. At roughly $75,000 for each person-year, the application cost $750,000. The system was deployed by the second quarter of 1990. Currently, it is used in 3 of 6 engineering departments that are involved with some phase of fluid system construction. The number of users makes up approximately 2.5 percent of the total engineering force. To date, the system has been used on more than 200 atypical nonconformances. The return during the first year of use is estimated to be $240,000, based on the following calculations: Some 20,000 nonconformances were processed. The manual resolution of nonconformances takes 3 hours for each nonconformance, on the average. At an hourly rate of $40, this figure calculates to $2,400,000. Overall, the system reduced the average time to process a nonconformance by about 10 percent, as estimated by current users and management. Thus, the estimated return for the first year of deployment is $240,000. This figure does not reflect additional savings realized from reduced scheduling delays, nor does it reflect other benefits derived from the internal research and development and manufacturing and production engineering projects. For example, the internal research and development project resulted in proposals of AI solutions for other problems in ship manufacturing and command and control systems (in progress or under evaluation), and the early trial development work of the manufacturing and production engineering project was used as a resource for applications suited for rule-based systems.

Generally, the case-based reasoning system demonstrated an increasing robustness and flexibility that was missing in the earlier rule-based systems. Engineers appear to be more comfortable with case-based reasoning techniques than with rule-based techniques. A general lesson learned from our efforts is that an engineer is more likely to use *X* if s/he understands *X*. In contrast with the rule-based systems, the knowledge base for the case-based reasoning system is compiled and maintained by users, and the knowledge base is automatically updated during use. The current task of the manufacturing and production engineering project is to deploy similar systems for electric and structural systems and investigate coupling the systems with existing databases. As the systems become more widely deployed and as the case libraries expand with use, we hope to see our return increase exponentially.

References

Alterman, R. 1986. An Adaptive Planner. In Proceedings of the Fifth National Conference on Artificial Intelligence, 65–69. Menlo Park, Calif.: American Association for Artificial Intelligence.

Ashley, K., and Rissland, E. 1988. A Case-Based Approach to Modeling Legal Expertise. *IEEE Expert* 3(3): 70–76.

Bain, W. 1986. A Case-Based Reasoning System for Subjective Assessment. In Proceedings of the Fifth National Conference on Artificial Intelligence, 523–527. Menlo Park, Calif.: American Association for Artificial Intelligence.

Brown, S. 1990. Nonconformance Resolution Using Case-Based Reasoning, MRR-EB90-046, General Dynamics Electric Boat Division, Groton, Connecticut.

Carbonell, J. 1983. Learning by Analogy: Formulating and Generalizing Plans from Past Experience. In *Machine Learning: An Artificial Intelligence Approach*, eds. R. Michalski, J. Carbonell, and T. Mitchell, 137–161. San Mateo, Calif.: Morgan Kaufmann.

Daube, F., and Hayes-Roth, B. 1989. A Case-Based Mechanical Redesign System. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 1402–1407. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Davies, T., and Russell, S. 1987. A Logical Approach to Reasoning by Analogy. In Proceedings of the Tenth International Joint Conference on Artificial Intelligence, 264–270. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Gentner, D. 1983. Structure-Mapping: A Theoretical Framework for Analogy. *Cognitive Science* 7: 155–170.

Hammond, K. 1986. CHEF: A Model of Case-Based Planning. In Proceedings of the Fifth National Conference on Artificial Intelligence, 267–271. Menlo Park, Calif.: American Association for Artificial Intelligence.

Huhns, M., and Acosta, R. 1988. ARGO: A System for Design by Analogy. *IEEE Expert* 3(3): 53–68.

Kedar-Cabelli, S. 1986. Purpose-Directed Analogy: A Summary of Current Research. In *Machine Learning: A Guide to Current Research*, eds. T. Mitchell, J. Carbonell, and R. Michalski, 81–85. Boston: Kluwer.

Kolodner, J.; Simpson, R. L.; and Sycara-Cyranski, K. 1985. A Process Model of Case-Based Reasoning in Problem Solving. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence,

284–290. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Koton, P. 1988. Reasoning about Evidence in Causal Explanations. In Proceedings of the Eighth National Conference on Artificial Intelligence, 256–261. Menlo Park, Calif.: American Association for Artificial Intelligence.

Lewis, L. 1989. AI Technology Development Year-End Report, GDEB-ERR-EB89-024, General Dynamics Electric Boat Division, Groton, Connecticut.

Riesbeck, C., and Schank, R. 1989. *Inside Case-Based Reasoning.* Hillsdale, N.J.: Lawrence Erlbaum.

Rissland, E., and Ashley, K. 1986. Hypotheticals as Heuristic Device. In Proceedings of the Fifth National Conference on Artificial Intelligence, 289–297. Menlo Park, Calif.: American Association for Artificial Intelligence.

Rissland, E., and Skalak, D. 1989a. Case-Based Reasoning in a Rule-Governed Domain. In Proceedings of the Fifth IEEE Conference on AI Applications, 46–53. Washington, D.C.: IEEE Computer Society.

Rissland, E., and Skalak, D. 1989b. Combining Case-Based and Rule-Based Reasoning: A Heuristic Approach. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 524–530. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Simoudis, E., and Miller, J. 1990. Validated Retrieval in Case-Based Reasoning. In Proceedings of the Eighth National Conference on Artificial Intelligence, 310–317. Menlo Park, Calif.: American Association for Artificial Intelligence.

Slade, S. 1991. Case-Based Reasoning: A Research Paradigm. *AI Magazine* 12(1): 42–55.

Stottler, R.; Henke, A.; and King, J. 1989. Rapid Retrieval Algorithms for Case-Based Reasoning. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 233–237. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.