Categorizing Case-Base Maintenance: Dimensions and Directions*

David B. Leake and David C. Wilson

Computer Science Department
Lindley Hall 215, Indiana University
Bloomington, IN 47405, U.S.A.
{leake,davwils}@cs.indiana.edu

Abstract. Experience with the growing number of large-scale CBR systems has led to increasing recognition of the importance of case-base maintenance. Multiple researchers have addressed pieces of the case-base maintenance problem, considering such issues as maintaining consistency and controlling case-base growth. However, despite the existence of these cases of case-base maintenance, there is no general framework of dimensions for describing case-base maintenance systems. Such a framework would be useful both to understand the state of the art in case-base maintenance and to suggest new avenues of exploration by identifying points along the dimensions that have not yet been studied. This paper presents a first attempt at identifying the dimensions of case-base maintenance. It shows that characterizations along such dimensions can suggest avenues for future case-base maintenance research and presents initial steps exploring one of those avenues: identifying patterns of problems that require generalized revisions and addressing them with lazy updating.

1 Introduction

The growing use of CBR applications has brought with it increased awareness of the importance of case-base maintenance (CBM). Large-scale CBR systems are becoming more prevalent, with case library sizes ranging from thousands (e.g., Cheetham 1997, Kitano & Shimazu 1996) to millions of cases (Deangdej *et al.* 1996). Large case-bases raise concerns about the utility problem for case retrieval (Francis & Ram 1993; Smyth & Cunningham 1996), underlining the potential need to control case-base growth through case deletion policies (Smyth & Keane 1995). Even for smaller case-bases, the difficulties raised by distributed case collection (Borron, Morales, & Klahr 1996) and the vagaries of real-world data raise concerns about the consistency and accuracy of case knowledge and show the need for strategies to address such problems (Racine & Yang 1997).

*The authors' research is supported in part by NASA under award No NCC 2-1035. This paper appears in Smyth and Cunningham, Eds., *Advances in Case-Based Reasoning: Proceedings of EWCBR-98*, Springer-Verlag, 1998. Copyright ©1998 Springer-Verlag. This paper is not camera-identical to the published version.

Despite a number of projects illuminating these issues for particular CBM systems and tasks, there is currently no common framework to guide a more general study of case-base maintenance. Such a framework would be useful for understanding the state of the art in case-base maintenance, illuminating current practice and facilitating the comparison of particular approaches, as has already proven useful for studying case adaptation (Hanney *et al.* 1995; Voß 1996). Moreover, given the early state of CBM research, a set of dimensions for categorizing case-base maintenance methods can help to identify problems and opportunities for study, suggesting points of exploration in the space of possible CBM systems.

This paper presents a first step in the direction of a general CBM framework. It proposes a set of dimensions for describing CBM, uses them to characterize sample CBM policies, discusses the relationship of CBM to the revision of other CBR *knowledge containers* (Richter 1995), and highlights points for investigation suggested by the framework. We make no claim of providing a final taxonomy or a complete summary. Nevertheless, we believe the dimensions provide a useful way to describe central aspects of current practice in CBM, and they have led us to identify opportunities for new case-based maintenance approaches. After describing the framework, we sketch how one of these opportunities is explored in CBMatrix, a case-based "intelligent component" (Riesbeck 1996) under development to aid in using problem-solving environments for scientific computing.

2 Defining Case-Base Maintenance

We define case-base maintenance as the process of refining a CBR system's case-base to improve the system's performance:

> Case-base maintenance implements policies for revising the organization or contents (representation, domain content, accounting information, or implementation) of the case-base in order to facilitate future reasoning for a particular set of performance objectives.

Note that this definition considers the information defining an indexing scheme to be an intrinsic organizational component of the case-base itself. Thus case-base maintenance may involve revising indexing information, links between cases, or other organizational structures and their implementations.

Maintaining case-base contents may affect a single case or multiple cases. It may revise the case representations used (e.g., changing the predicates used to describe domain features); may revise either domain information in the case-base (e.g., correcting an erroneous feature in a case or adding or deleting an entire case) or "accounting" information (e.g., changing information about how frequently a case has been accessed); or may revise how case representations are implemented (e.g., changing from lists to feature-vectors). Thus maintenance of case-base contents may revise the case-base at the implementation level, representation level, or the knowledge level (cf. Dietterich 1986).

Our definition of maintenance implicitly includes policies for performing CBM indirectly, by revising the maintenance policies themselves. In section 4, we give a brief description of one approach to such "meta-maintenance."

Performance objectives provide criteria for evaluating the internal behavior and task performance of a particular CBR system for a given initial case-base and sequence of problems solved. The performance objectives may be quantitative (e.g., achieving particular problem-solving time or limiting case-base size), or qualitative (e.g., to extend system competence). Smyth (1998) provides compelling arguments for the importance of shaping maintenance policies according to a complete set of performance objectives. Of course, performance objectives may change over time to reflect varying external circumstances, which may necessitate changing (maintaining) maintenance policies as well.

3 A Framework for Describing CBM Policies

The goal of a categorization scheme for case-base maintenance is threefold. First, by identifying classes of similar maintenance approaches, such a categorization scheme can shed light on the state of current practice in the field, increasing understanding of current CBM approaches. Second, mapping out the space of candidate approaches helps identify parts of the space that have not been addressed in previous work; these gaps in turn suggest research opportunities. Third, a categorization scheme for maintenance approaches is a first step towards cataloging the approaches that are most appropriate for particular performance goals.

Our framework categorizes case-base maintenance approaches in terms of *case-base maintenance policies* that determine when and how a CBR system performs case-base maintenance. Maintenance policies are described in terms of how they gather data relevant to maintenance, how they decide when to trigger maintenance, the types of maintenance operations available, and how selected maintenance operations are executed.

In the framework, *Data collection* gathers, synthesizes, and distills the data about the case base and about system processing; this is the information that will be used to determine whether maintenance operations should be performed. *Triggering* takes this information as input, makes the decision whether maintenance is needed, and selects maintenance actions from a range of possible *Operation types. Execution* describes how the selected revisions are actually applied to the case-base.

Descriptions generated using the framework characterize basic combinations of policy attributes. A single CBR system may include multiple maintenance policies, each one implementing a different part of the system's overall maintenance agenda.

Data collection: Data collection gathers information about individual cases, about the case base in part or as a whole, and/or about the overall processing behavior of the CBR system. Data collection about individual cases might record the number of times a case has been successfully used or the number of times it has failed. Data collection about the case base as a whole could involve, for example, monitoring the size of the case base. Data collection about processing might involve noting clusters in input problems or input problems that the system is unable to solve successfully.

Type of data: None, Synchronic, or Diachronic: There are three approaches to collecting and analyzing data to decide when case base maintenance is needed. The simplest is to do no collection at all. A policy with no data collection makes maintenance decisions independently of the present or past state of the case base. As such, this type of policy is referred to as *non-introspective*. For example, a CBR system that updates its case-base by unconditionally adding a case each time it adapts a retrieved case to new needs would need no data collection. This is the approach of most CBR systems.

More sophisticated reasoning is enabled by considering a snapshot of the current case-base in part or as a whole. Examination of this information can determine, for example, whether a case is worth adding to a case-base because it increases the competence of the CBR system, or whether a solution can be discarded without affecting competence (Smyth & Keane 1995). Policies that consider snapshot information are called *synchronic*.

The most informative approach is to collect data over time, over a sequence of snapshots, in order to identify trends in how case-base contents and usage are changing. Policies that consider changes in the case-base over time are called *diachronic*. For example, a policy that gathered information about trends in retrieval times, to identify the onset of utility problems, would be diachronic. Because synchronic and diachronic collection examine the internal state of the case base, both are referred to as *introspective*.

Timing: Periodic, Conditional, or Ad Hoc: A maintenance policy must specify when data collection is performed. In our framework, there are three possibilities. *Periodic* timing happens at a set frequency with respect to the CBR cycle. For example, data collection might be performed after each problem-solving cycle. Periodic timing that happens every cycle is termed *continuous*. *Conditional* data collection is performed in response to a well-defined but non-periodic condition. For example, analysis might be triggered whenever the number of cases in the case library reaches a particular threshold (Smyth & Keane 1995). *Ad hoc* timing happens under ill-defined conditions determined externally to the CBR system.[1] Examples of ad hoc timing are user-initiated tests on the case base to determine whether maintenance is needed or a domain expert's decision to add new cases regardless of the case base contents.

Integration: On-line or Off-line: Data collection may operate *on-line*, during the course of an active reasoning episode, or *off-line*, during a pause in reasoning, such as waiting for user input or when idle between reasoning episodes. The

[1] This category name in no way implies that the choice is ill-considered; simply that it is not under control of the policy.

choice between on-line and off-line processing may affect the resources that can be devoted to the analysis process, making it important for determining whether a policy is appropriate for time-constrained processing.

Triggering: The results of data analysis serve as input for determining whether case-base maintenance is necessary. Both the *timing* and *integration* dimensions discussed previously apply to this step as well. Strategy selection can be done periodically, conditionally, or on an ad hoc basis, and on-line or off-line.

Conditional triggering can be subdivided into three classes depending on the conditions that determine whether maintenance is triggered: *space-based* (e.g., filling a limited amount of case storage), *time-based* (e.g., retrieval time exceeding a threshold), or *result-based* (e.g., the system failing to solve a given problem or the wrong case being retrieved).

Operation types: Different maintenance policies revise different types of information (the *target type*) at different levels (the *revision level*).

Target type: Revision operations can focus on four types of targets: *Indexing structures*, *domain contents*, *accounting information*, and, as will be described in section 4, *maintenance policies* themselves.

Revision level: Revision operations can make revisions at three levels: The *implementation level* (e.g., changing an indexing structure from a list to a D-tree when the case-base exceeds a certain size or changing case representations from lists to vectors), the *representation level* (e.g., reconciling inconsistent feature names or case formats in cases that come from different sources), or the *knowledge level* (e.g., correcting an erroneous feature value, generalizing case values, or adding or deleting cases).

Finer-grained characterizations of operator types are of course possible (e.g., Heister and Wilke (1998) describe a set of atomic maintenance operations). However, as with the rest of the categorization scheme, we have used higher-level categories to facilitate cross-system comparisons of major characteristics.

Execution: Execution is characterized by the timing of maintenance operations and their integration with other system processing. Execution timing is described using the timing dimension previously described for data collection (periodic, conditional, or ad hoc); timing may also be "none" for systems with no execution. For example, a maintenance policy may simply inform a maintainer that maintenance is needed without making changes (none); changes may be made on a regular basis (periodic); changes may be held for batch updating when enough cases are accumulated (conditional); or changes may be held for when an expert is available (ad hoc). Likewise, execution integration is described as on-line or off-line depending on whether maintenance operations are performed during or between reasoning episodes.

Scope of Maintenance: Broad or Narrow: A given operation may be applied to few or many items in the case base. Operations that affect a single case or a small subset of the case-base have *narrow* scope, and operations that affect a large subset or the entirety of the case base have *broad* scope. This dimension is especially useful when characterizing resource-bounded processing.

3.1 Examples

To illustrate the use of the framework we apply it to a sampling of CBM approaches, beginning with a few simple examples. For reasons of space, we emphasize two parts of the CBM framework that we consider particularly useful for describing current CBM systems: the type of data collected and how maintenance policies are executed. Figure 1 summarizes the described approaches along these dimensions.

Policies targeting domain content: The standard learning of CBR problem-solving systems (always adding each new case to the case base), is designated CBR_1. No data analysis is performed—the new case is recorded without considering the existing contents of the case base—so it is non-introspective. Because learning happens during each reasoning cycle, this policy is continuous (periodic) and on-line. Because only a single case is added, the scope of change is narrow.

Another common CBR method (CBR_2) involves a non-learning system maintained by a domain expert who sometimes adds a variable number of new cases. For this method, we presume no analysis of the existing case-base, so the maintenance policy is non-introspective. Because the timing of the updates depends on the expert's external decision, the timing is ad hoc. Because the cases are added manually outside of normal processing, the integration is off-line. Because the number of cases can be small or large, the scope varies from narrow to broad.

Shimazu and Takashima describe an enhanced version of the CARET system (S&T) that identifies discontinuities in a case-base (Shimazu & Takashima 1996). That system uses synchronic data collection; it retrieves a set of "Maybe Similar Cases" (MSCs), chooses a single best "Base Case"(BC), and classifies as "discontinuous" any remaining MSCs whose suggestions differ from the BC by more than a given threshold, identifying them as potential candidates for maintenance. However, the system does not execute revisions.

Smyth and Keane (1995) describe a competence-preserving approach to case deletion, which specifies a case utility hierarchy in terms of coverage and reachability. When the number of cases in the case-base exceeds the "swamping limit," their "footprint-utility deletion" strategy selects candidates for deletion based on the utility hierarchy. Because the hierarchy is defined with respect to the current state of the case-base, the policy is synchronic. Because maintenance is triggered in response to the current size of the case-base, timing is conditional. Smyth and Keane describe this mechanism being applied either to small numbers of cases during processing, using a heuristic method of utility evaluation ($\text{S\&K}_2$, on-line and narrow) or to large numbers of cases with full analysis outside of the reasoning cycle ($\text{S\&K}_1$, off-line and broad).

Racine and Yang (1997) describe policies for identifying redundant cases ($R\&Y_1$), and for identifying inconsistent cases ($R\&Y_2$). Both policies rely on an analysis of the current state of the case-base, so they are synchronic. Both are applied to the case-base as a whole when desired by a case-base maintainer, so they are broad, ad hoc and off-line.

Watson (1997) presents a set of guidelines for human case-base maintainers (W) that involve performing periodic tests on the entire case-base. This policy can be described as having synchronic analysis, ad-hoc timing, off-line execution, and narrow or broad scope.

The second of the coordinated trend-based policies to be presented in Section 5 ($L\&W_2$) makes narrow changes to cases in a lazy manner, on-line, according to a maintenance rule installed by the system. It changes only those case that have not yet been updated, so it is conditional.

Policies targeting indices: A number of classification systems using IBL and related techniques (IBL) include policies for eliminating noisy and redundant instances from a set of training examples (cases). These systems generalize a case-base either explicitly, by merging cases with similar coverage (e.g., Domingos 1995) or implicitly, by choosing a smaller, representative subset of cases (e.g., Aha, Kibler & Albert 1991). Such policies typically consider a static set of cases (synchronic), are user-initiated (ad hoc), perform execution off-line, and are applied to the entire training set (broad). Because case features (other than the category) are only used as indices, we view their generalizations as revising indexing information. When IBL systems remove noisy instances or remove a class entirely, their target is domain content.

Fox and Leake (1995) describe a policy (F&L) that triggers index revision for plan cases in response to plan failures. This policy considers snapshot information about execution (synchronic), is executed conditionally, is performed on-line, and revises indices in the entire case-base (broad scope).

Aha and Breslow (1997) describe an index revision method (A&B) that considers an entire case-base in response to an external request. This policy has no data collection, ad hoc activation timing, off-line integration, and broad scope.

Racine and Yang (1997) describe a third policy for deriving and updating indices of unstructured cases ($R\&Y_3$). Like their other policies, this policy is synchronic, broad, off-line, and has ad-hoc execution.

Policies targeting maintenance policies: The first policy presented in section 5 ($L\&W_1$) uses diachronic information to trigger changes when a potentially important trend is detected. It performs a narrow change—adding a new maintenance policy.

4 Meta-Maintenance by Lazy CBM

When a CBR system retrieves a case and adapts it to fit a new situation, CBM normally stores the result of adaptation as a new case and leaves the original

				Data Collection		
				Type of Data		
	Activation Timing	*Integration Type*	*Scope of Changes*	None	Synchronic	Diachronic
Execution	Periodic	On-line	Broad			
			Narrow	CBR$_1$		
		Off-line	Broad			
			Narrow			
	Conditional	On-line	Broad		F&L	
			Narrow	L&W$_2$	S&K$_2$	L&W$_1$
		Off-line	Broad		S&K$_1$	
			Narrow			
	Ad hoc	On-line	Broad			
			Narrow			
		Off-line	Broad	CBR$_2$	A&B, IBL, W, R&Y$_{1,2,3}$	
			Narrow	CBR$_2$	W	
	No Execution				S&T	
				Non-Introspective	*Introspective*	

Fig. 1. Sample CBM approaches placed along major dimensions

case unchanged. However, if there are defects in the old case, case-base maintenance can simultaneously revise the old case and re-store it in its updated form. This approach updates old cases in a "lazy" manner as they are applied to new situations. It is driven by a process similar to case adaptation, but whose aim is to repair a problem in an old case rather than to fit that case to a specific new situation. This allows expensive updates to be performed only on the portion of the case base that is actually being used, decreasing update effort while still allowing future processing of frequently-used cases to start from the updated versions. Thus when a change must be applied throughout the case-base, a CBM system can either (1) make that change to all old cases simultaneously, when it next performs overall maintenance, or (2) generate a maintenance rule to update each case that is retrieved, when it is retrieved (and before it is applied to the new situation). Installation or revision of these maintenance rules can be viewed as a form of "meta-maintenance," maintaining the system's maintenance knowledge. Note that both approaches achieve a broad change, but that the second does so by implementing two policies with narrow scope, making it preferable in time constrained circumstances.

With a lazy updating scheme, cases that are obsolete must be distinguished from cases that have been updated. When a sufficiently large proportion of the retrieved cases have already been modified by a maintenance rule, it may be possible to abandon the rule—the case base may have been sufficiently modified for the problems the system tends to encounter.

In rapidly-changing domains, especially if application of maintenance rules is inexpensive, it may be preferable never to update the stored cases, instead composing old maintenance rules into new ones to obtain the desired net changes (e.g., to take compound inflation into account). Such a method also facilitates retraction of invalid updates: flawed maintenance rules can be retracted without making any changes to existing cases.

5 Trends and Lazy Maintenance: An Example from CBR for Scientific Computing

In scientific computing, problem solving environments (PSEs) provide scientists with a framework of integrated problem-solving tools that they can easily configure and apply to problems that arise in their particular task domains. Because effective solution strategies depend on making good choices about the organization and configuration of these tools, considerable expertise may be needed to achieve full benefit from the tools provided by a PSE. However, it is often difficult to capture principles guiding tool and parameter selection. Consequently, CBR methods to guide tool selection, organization, and application have the potential to play a valuable role in PSEs. The CBMatrix project investigates CBR and CBM issues arising in the context of CBR components within a scientific PSE, the Linear System Analyzer (LSA) (Gannon *et al.* 1998), which is aimed at aiding the solution of sparse linear systems. Given a scientific computing problem to solve within the LSA, CBMatrix retrieves prior cases that suggest computational methods and parameters for solving the problem efficiently (e.g., the data structures to use to achieve the highest megaflop performance rating).

The PSE advisory task requires the management of substantial case libraries in the face of unreliable information, limited feedback, limited storage, and changing external circumstances. A particularly acute issue concerns how to revise the case-base to improve performance when classes of problems change (e.g., when a scientist begins to apply the scientific computing system to a series of problems with different characteristics from those for which the case base was built) or when changes in the external environment affect the quality of the advice offered by a pre-existing case-base (e.g., if the scientist runs CBMatrix on one set of problems, on one computer, to build a library of advice on methods for solving those problems, and then buys a new computer with hardware that renders some of the prior advice obsolete). Thus this domain requires addressing not only the maintenance issues involved in dealing with potentially noisy and unreliable data (e.g., because results depend on the external load on the machine), but also on addressing questions about how to maintain a case-base when new hardware requires systematic changes in the recommendations the CBR system provides.

The CBMatrix system implements two maintenance policies that together result in a lazy update of the case-base by a "pre-adaptation" revision of retrieved cases. The first policy installs a new maintenance rule when needed, as described in the previous section. This policy is triggered by diachronic analysis of successive snapshots of the case-base as new situations are processed, in order to recognize changes in machine characteristics. The data collection process for this policy monitors the predictions made by retrieved cases about the expected performance of the most appropriate data structure for solving a given system. If the processing results in performance that is either significantly worse (unexpected failure) or significantly better (unexpected success), the result is added to a data set that is analyzed for trends in performance. Individual fluctuations might be due to processing loads, etc., while consistent trends suggest a more durable change.

The number and magnitude of the unexpected successes or failures with respect to time (measured in numbers of reasoning episodes/cycles) define a trend in performance anomalies that can indicate a changing trend in the linear system processing results (e.g., because the computer being used to solve the problems has been upgraded). Once a trend has achieved a certain level of activation, this maintenance policy installs the second maintenance policy, a new maintenance rule to adjust subsequent predictions (e.g., if there were a trend for predictions to be 20% pessimistic, the rule would adjust predictions upwards on each retrieved case that had not yet been adjusted). This is a simple approach to a problem that is in general very complex, but it appears practical for this type of change and shows the benefit of considering diachronic information when triggering maintenance.

6 Maintenance and Overlapping Knowledge Containers

The multiple knowledge containers of CBR overlap; knowledge available in one can replace missing knowledge in another (Richter 1995). Likewise, the effects of maintenance to one knowledge source may be equivalent to maintenance on another. For example, the same overall effects on system accuracy might be achieved by case-base reorganization—which we consider part of case-base maintenance—or by adjustment of the similarity measure—which we consider external to CBM. Although our framework focuses only on case-base maintenance, in general CBM can be viewed as part of the larger task of CBR system maintenance (e.g., Heister & Wilke 1998).

7 Future Work

Many CBM issues remain to be investigated. Because, to our knowledge, the role of usage trends in guiding maintenance has not yet been explored in other research, we consider it an especially promising area. The very simple trend-based maintenance described in the previous section has application to a particularly well-behaved type of change in the case-base that appears in other contexts as well (e.g., updating old prices based on inflation, for real-estate appraisal) but would fail to apply to more subtle trends that would require more sophisticated methods.

Another form of trend information that might be exploited, for example, is examination of patterns in the types of problems that are being solved—to identify “hot spots” in the problem space and identify subsets of the case-base to be consulted first, while (if storage were limited), less useful cases could be archived. Racine and Yang (1997) observe that recent cases may be likely to be useful; trend analysis could provide other types of suggestions for which cases should be most accessible.

A long-term goal of characterizing maintenance policies is to combine these characterizations with descriptions of the tasks, domains, and performance objectives for which particular policies are likely to be appropriate, to help guide policy selection decisions when developing CBR systems.

8 Conclusion

This paper presents an initial framework for characterizing case-base maintenance policies. It presents basic dimensions for CBM policies in terms of three subprocesses—data collection, triggering, and execution—and characterizes key design choices in terms of those dimensions. Factors considered include the type of information collected, timing, and integration of data collection; the timing and integration of maintenance triggering; the types of maintenance operations used; and the timing, integration, and scope of maintenance execution. The paper demonstrates the use of this framework to describe sample approaches to CBM, and shows the potential of the framework to suggest areas for study by discussing a simple implementation of a diachronic case-base maintenance policy. Further examination of the framework—and of the case-base maintenance task—is clearly needed, and our most immediate task is to apply it to more data from current practice. Nevertheless, we hope that this paper will spark discussion and further investigation, both of the practice of case-based maintenance and of issues and opportunities for new CBM approaches.

References

Aha, D., and Breslow, L. 1997. Refining conversational case libraries. In *Proceedings of the Second International Conference on Case-Based Reasoning*, 267–278. Berlin: Springer Verlag.

Aha, D.; Kibler, D.; and Albert, M. 1991. Instance-based learning algorithms. *Machine Learning* 6:37–66.

Borron, J.; Morales, D.; and Klahr, P. 1996. Developing and deploying knowledge on a global scale. In *Proceedings of the Thirteenth National Conference on Artifical Intelligence*, volume 2, 1443–1454. Menlo Park, CA: AAAI Press.

Cheetham, W., and Graf, J. 1997. Case-based reasoning in color matching. In *Proceedings of the Second International Conference on Case-Based Reasoning*, 1–12. Berlin: Springer Verlag.

Deangdej, J.; Lukose, D.; Tsui, E.; Beinat, P.; and Prophet, L. 1996. Dynamically creating indices for two million cases: A real world problem. In Smith, I., and Faltings, B., eds., *Advances in case-based reasoning*, 105–119. Berlin: Springer Verlag.

Dietterich, T. 1986. Learning at the knowledge level. *Machine Learning* 1:287–316.

Domingos, P. 1995. Rule induction and instance-based learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 1226–1232. San Francisco, CA: Morgan Kaufmann.

Fox, S., and Leake, D. 1995. Modeling case-based planning for repairing reasoning failures. In *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms*, 31–38. Menlo Park, CA: AAAI Press.

Francis, A., and Ram, A. 1993. Computational models of the utility problem and their application to a utility analysis of case-based reasoning. In *In Proceedings of the Workshop on Knowledge Compilation and Speed-Up Learning.*

Gannon, D.; Bramley, R.; Stuckey, T.; Villacis, J.; Balasubramanian, J.; Akman, E.; Breg, F.; Diwan, S.; and Govindaraju, M. 1998. Component architectures for distributed scientific problem solving. *IEEE CS&E.* In press.

Hanney, K.; Keane, M.; Smyth, B.; and Cunningham, P. 1995. What kind of adaptation do CBR systems need? a review of current practice. In *Proceedings of the Fall Symposium on Adaptation of Knowledge for Reuse.* AAAI.

Heister, F., and Wilke, W. 1998. An architecture for maintaining case-based reasoning systems. In Cunningham, P.; Smyth, B.; and Keane, M., eds., *Proceedings of the Fourth European Workshop on Case-Based Reasoning.* Berlin: Springer Verlag. In press.

Kitano, H., and Shimazu, H. 1996. The experience sharing architecture: A case study in corporate-wide case-based software quality control. In Leake, D., ed., *Case-Based Reasoning: Experiences, Lessons, and Future Directions.* Menlo Park, CA: AAAI Press. 235–268.

Racine, K., and Yang, Q. 1997. Maintaining unstructured case bases. In *Proceedings of the Second International Conference on Case-Based Reasoning*, 553–564. Berlin: Springer Verlag.

Richter, M. 1995. The knowledge contained in similarity measures. Invited talk, the First International Conference on Case-Based Reasoning, Sesimbra, Portugal.

Riesbeck, C. 1996. What next? The future of CBR in postmodern AI. In Leake, D., ed., *Case-Based Reasoning: Experiences, Lessons, and Future Directions.* Menlo Park, CA: AAAI Press.

Shimazu, H., and Takashima, Y. 1996. Detecting discontinuities in case-bases. In *Proceedings of the Thirteenth National Conference on Artifical Intelligence*, volume 1, 690–695. Menlo Park, CA: AAAI Press.

Smyth, B., and Cunningham, P. 1996. The utility problem analyzed: A case-based reasoning perspective. In Smith, I., and Faltings, B., eds., *Advances in case-based reasoning*, 392–399. Berlin: Springer Verlag.

Smyth, B., and Keane, M. 1995. Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 377–382. Montreal: IJCAI.

Smyth, B. 1998. Case-base maintenance. In *Proceedings of the Eleventh International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems.*

Voß, A. 1996. Principles of case reusing systems. In Smith, I., and Faltings, B., eds., *Advances in case-based reasoning*, 428–444. Berlin: Springer Verlag.

Watson, I. 1997. *Applying Case-Based Reasoning: Techniques for Enterprise Systems.* San Francisco: Morgan Kaufmann.