Six Steps in Case–Based Reasoning: Towards a maintenance methodology for case–based reasoning systems

Thomas Roth–Berghofer[1] and Ioannis Iglezakis[2]

[1] tec:inno GmbH,
Sauerwiesen 2, 67661 Kaiserslautern, Germany
thomas.roth-berghofer@tecinno.com
[2] DaimlerChrysler AG, Research & Technology, FT3/AD,
P.O. Box 2360, 89013 Ulm, Germany
ioannis.iglezakis@daimlerchrysler.com

Abstract. The growing demand for maintenance of case–based reasoning systems has lead to intensive research. Many aspects of maintenance were examined in the recent years, mainly concentrating on maintaining the case–base. In this paper, we look at foundational issues regarding the maintenance of case–based reasoning systems. We revisit the traditional four step process model and discuss some shortcomings regarding maintenance. Based upon these, we discuss the extension of the four step cycle by the two steps *review* and *restore*, and relate already developed maintenance methods to the six step process model.

1 Introduction

Case–based reasoning (CBR) is a knowledge–based problem solving technique that is based on reuse of past experience. Today, many CBR systems are available and successfully running. From the experience of building CBR systems the INRECA methodology was developed [3]. This methodology describes how to design and implement CBR solutions for successful application in real–world industrial contexts. But this methodology does not answer the questions of how to maintain the developed solution.

Maintenance of case–based reasoning systems is a hot topic in CBR research and application. Many aspects of case–based reasoner maintenance have been explored mainly concentrating on case base maintenance [7].

Software system maintenance, in general, has to deal with three types of maintenance [20]: *corrective*, *adaptive*, and *perfective* maintenance. And — to add another level of difficulty — in knowledge–based systems we do not only have to deal with the maintenance of the case–based reasoning application itself we also have to deal with the maintenance of the system's knowledge. But in this paper we concentrate on the maintenance of the knowledge.

Corrective maintenance is the most basic type of maintenance. It is used in order to correct erroneous system behavior. Examples are the deletion or adaptation of incorrect cases, or adjustments of similarity measures which lead to wrong solutions. Adaptive maintenance adjusts the system to environmental changes, e.g., it adjusts the system to

new products in a product database. Perfective maintenance looks at processing inefficiencies, performance enhancements, and the maintainability of the system. Especially looking at performance enhancements is a main topic in CBR research.

But a methodology is still missing which covers these types of maintenance and provides tasks and methods to deal with changes of the CBR system and its environment systematically. With this work we try to make a step into that direction.

This paper is organized as follows. In the next section, we provide a short overview over related work. Then, we revisit the four step process model of Aamodt and Plaza [2] and describe major weaknesses regarding the maintenance of case–based reasoning systems. In section 4, we describe a six step process model, and a decomposition into tasks and methods. We close the paper with some concluding remarks and issues for future work.

2 Related work

In 1994 Aamodt and Plaza defined a well received four step view on the process of case–based reasoning [2]. The four steps *retrieve*, *reuse*, *revise*, and *retain* — called the four REs — are widely acknowledged.

Enhancing the CBR process model is not a very new idea. Göker and Roth-Berghofer [5] described organizational, management–related, and technical processes regarding the development and utilization of a help–desk tool. They extended the CBR process by separating it into an *application* and a *maintenance* cycle which allows to handle changes of the system explicitly.

Their application cycle takes place each time a user solves a problem with the CBR system. It consists of the three steps retrieve, reuse, and revise. If there is a new case to be stored the new case is put to use during the *recycle* step. Those cases are marked as *unconfirmed* and sent to the maintenance cycle.

Their maintenance cycle consists of the steps *retain* and *refine*. During the retain step a CBR engineer checks the quality of the new cases and during the refine step the engineer performs steps to increase the quality of the knowledge. But the refine step was not specific enough to reflect processes necessary to define the maintenance phase.

Of course, the application and maintenance phases are interwoven. There are classes of CBR systems where cases are learned after the retrieval phase and there are many systems using only the application phase. An interesting way of controlled interaction of application and maintenance phase is the collaborative maintenance framework of Ferrario and Smyth [4] which was designed to facilitate, monitor, and control the incremental update of dynamic case–bases. Their approach automatically supports a distributed, interactive maintenance process — users are permitted to recommend case updates and the collaborative maintenance process ensures that these recommendations are properly reviewed and put into action.

Another enhancement of the original cycle resulted from discussions at the CBR workshop at IJCAI 1999 [21]. The participants saw the need for additional steps, too. They proposed the steps *review* and *reflect*. Their review step is inserted before the retain step and covers tasks to assess the quality of a single case before adding the case to the case base. Thus, this review step ensures that only interesting and valuable cases are

stored. Their second additional step reflects on feature weights, similarity metrics, case coverage, etc. In comparison to our approach, their review step only considers single cases without their relation to other cases in the case base, whereas their reflect step corresponds more to our review step.

Reinartz, Iglezakis, and Roth-Berghofer [14] developed this further. They propose an extended 6 step CBR cycle and discuss its two additional steps as part of the maintenance phase of the CBR process focusing their attention on case–base maintenance. In this paper, we look at foundational issues regarding the complete CBR system.

3 The four REs revisited

The framework of Aamodt and Plaza [2] for describing CBR methods and systems consists of two main parts: a process model of the main CBR steps, and a task–method structure.

The two models are complementary and represent two views on case–based reasoning. The first is a dynamic model that identifies the main subprocesses of a CBR cycle, their interdependencies and products. The second is a task–oriented view, where a task decomposition and related problem solving methods are described. Aamodt and Plaza identified and discussed important problem areas of CBR, and means of dealing with them. All task–decompositions are complete, i.e., the set of subtasks of a task are intended to be sufficient to accomplish the task.

The four processes *retrieve*, *reuse*, *revise*, and *retain* describe the general tasks in a case–based reasoner. They provide a global external view to what is happening in the system. The four tasks are decomposed into a hierarchy of CBR tasks the system has to achieve. This task–oriented view is suitable for describing the detailed mechanisms from the perspective of the case–based reasoner itself.

Aamodt and Plaza call the top level task of CBR *problem solving and learning from experience* which directly matches our two phases application and maintenance mentioned above. In the following, we will shortly revisit the four steps. We already distinguish between the two phases here. We close this section with a discussion of the shortcomings of this process model regarding maintenance.

3.1 Application Phase

The application phase consists of the three steps *retrieve*, *reuse*, and *revise*. These subprocesses are shown in figure 1 (Only the task–level is reproduced here from [2].).

In the first step the case–based reasoner retrieves the most similar case or cases. Then, it reuses the information and knowledge in that case and proposes a solution. If the solution is rejected the CBR system revises the proposed solution. This far the systems stays unchanged. If the revised solution is stored for later use during the retain step the case–based reasoner enters the maintenance phase.

Using the knowledge containers metaphor [16] the tasks (and methods to accomplish the tasks, respectively) affect different parts of the case–based reasoner, e.g., *identifying features* mainly involves the vocabulary to ‘understand’the problem within its

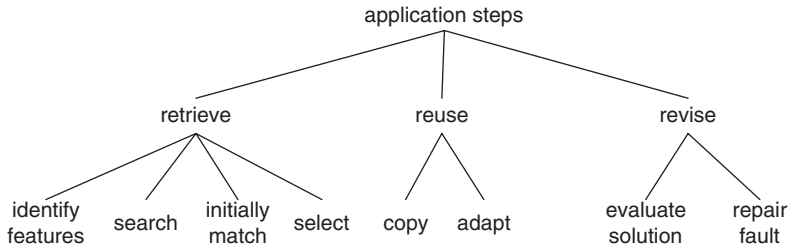

Fig. 1. Application steps and their task decomposition (adapted from [2])

context, and to solve the tasks *search* and *initially match* the similarity measures container is mainly used. Methods to accomplish these tasks may be to use k–d–trees [22] or case retrieval nets [11]. The *select* task, finally, also uses the similarity measures to rank the best cases which may involve the use of the case base container if the selection methods are knowledge–intensive and the index structures are not sufficient for this task. During the retrieve step no adaptation knowledge is used.

None of the steps of the application phase changes any of the knowledge containers of the case–based reasoner.

3.2 Maintenance Phase

The maintenance phase in the original CBR cycle consists of the retain step alone. During this phase the revised solution is stored in the case base.[1] The indexing structures to the newly added case are adapted, too. If no new case was constructed existing indexing structures are modified to improve the similarity assessment.

The retain step is the point where change is introduced into the CBR process.

3.3 Shortcomings

The four step CBR cycle is an adequate way to describe a running system. In this way the model provides a complete description. But regarding maintenance this model is insufficient in several aspects.

First, there is no explicit distinction between steps that do not change the case–based reasoning system and the step that does. This distinction between application and maintenance phase is important because changes of the system are not necessarily due to learning alone. Changes of the environment can impose changes on the knowledge containers. Dividing the processes into an application and a maintenance phase allows to view different versions of the case–based reasoner separately.

[1] Aamodt and Plaza state that *in CBR the case base is updated no matter how the problem was solved* ([2], p. 52). But this is not true for all CBR systems. It depends on the application domain. For example, in an electronic commerce scenario CBR may be used to find matching products [23] but there mostly will be no creation of products.

Learning in the four step model only means shifting knowledge from one knowledge container to another.[2]

Second, in the four step model no maintenance data is collected. To assess the quality of a CBR system performance data must be collected. Every step offers an opportunity to remember usage data, for example.

Third, because change is not explicitly handled in the four step process model there was no introspection designed to monitor system changes.

To remedy this shortcomings a six step process is proposed in [15] which will be described in the following section.

4 Six steps in case–based reasoning

The proposed six step process model consists of the original four steps *retrieve*, *reuse*, *revise*, and *retain*, and the two novel steps *review* and *restore*. Figure 2 shows the cycle with its separation into application and maintenance phase. To emphasize this two

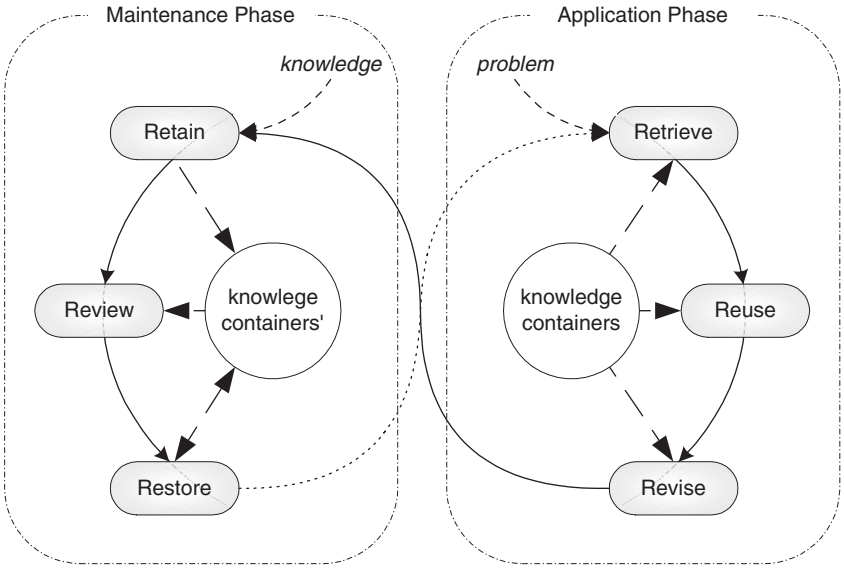

Fig. 2. The six RE cycle (adapted from [15])

phases two versions of the system are displayed. The knowledge containers of each phase are normally consecutive versions of each other (indicated by the arrows from revise to retain and from restore to retrieve).

[2] For example, [24] describes a knowledge light approach to learn adaptation knowledge from the other knowledge containers.

In the six step model changes initiated from outside of the CBR system can be modeled easily. For example, new cases may be added to the case base in two ways (depicted in figure 2 by the problem entering the system at the retrieve step and by other knowledge entering at the retain step). First, there may be new cases as results of the revise step. Second, new cases may come from outside. This often can be found in industrial CBR systems where cases are stored as documents or in databases. In regular intervals cases are brought into such CBR systems.

4.1 Enhancing the four steps

At every step of the original cycle information should be collected to support the maintenance process. Therefore, we proposed an enhancement of the case definition which reflects this. In general a case consists of two parts: the problem description and the solution.[3] In our view a case should consist of a third third part, called *quality information*. This part should collect all the data needed for review and restore.

Performance data already is collected in many systems to support maintenance (or to provide marketing figures). In the following we suggest some data to be collected at each step.

Retrieve The retrieve step provides information through its result, namely, how often a case is retrieved. Also, to improve performance the mean retrieval time may be stored. Maximum and minimum may also be of interest. Other usage information are the last update or last retrieval time. The queries may be logged here, too. In the maintenance phase the queries could be used to enhance the vocabulary, the similarity measures or the adaptation knowledge. They even may be used to seed new cases. Comparisons of retrieval results of different versions of the case–based reasoner regarding the stored queries may yield interesting insights into the behaviour of the system over time.

Reuse The reuse step adapts the retrieved cases to solve the given problem. An adaptation counter can count which of the retrieved cases could be adapted to the query and is therefore suggested as a possible solution.

Revise If the suggested solution is rejected this may be counted as well as how often the case has been accepted as a solution. This information can help to decrease the case base size because it may be more favorable to remove an often rejected case from the case base than accepted cases.

Retain This step adds adapted cases to the system. As discussed in [5] it would be advisable to mark those cases as *unconfirmed*. The user then has a choice between confirmed and unconfirmed cases. A second purpose of the retain step is to modify the similarity measures by modifying the indexing structures. Modifications like that should only be used by the case–based reasoner if there are possibilities to track the changes, or better measure the impact of those changes.

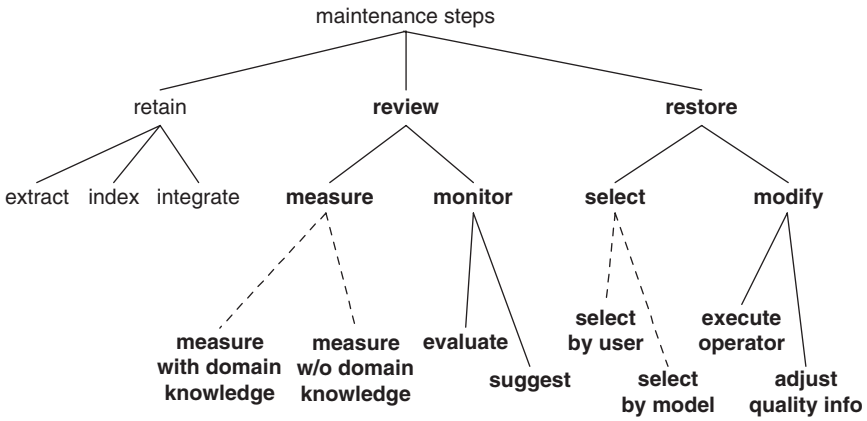

Fig. 3. Maintenance steps and their decomposition in tasks and methods

4.2 Review

The review step consists of two tasks: *measure* and *monitor*. Figure 3 shows the maintenance steps with their decomposition into subtasks (plain lines) and alternative methods (dashed lines). We adopted the task–method structure from [2] to emphasize the similarities and differences between the original cycle and our extensions. Besides, the task–method structure will serve as a basis for a maintenance methodology.

The review step considers the current state of the knowledge containers and assesses their quality. For this purpose, appropriate quality measures have to be selected which indicate the quality of the evaluated CBR system.

In [14] Reinartz, Iglezakis, and Roth-Berghofer define syntactical measures like correctness, consistency, uniqueness, minimality, and incoherence. We call them syntactical measures because they do not rely on domain knowledge (except correctness which, of course, is based on the underlying domain theory). Other questions which could be asked here are: Is the case, similarity measure, adaptation knowledge, or vocabulary (still) applicable? Are the cases, etc. (still) relevant?

Checking the vocabulary versus the case base can show what part of the domain model actually is used by the case base, thus indicating if there are regions in the vocabulary which are not covered by cases.

Semantical measures — measures using domain knowledge — are those like case base density or case base distribution [19]. Coverage and reachability are other well known criteria to assess case base quality [18]. If adaptation knowledge is available competence preserving strategies may be used to improve CBR system quality. If no adaptation knowledge can be used (or can not be used easily) a reduced version of coverage and reachability might be computed based on similarity measures alone.

[3] For a discussion of case types other than ordered pairs of problem and solution, refer to [12].

Leake and Wilson [10] examine the relationship between competence and performance, and discuss the goals and constraints that should guide addition and deletion of cases. They illustrate the importance of augmenting competence–based criteria with quantitative performance–based considerations, and propose a strategy for reflecting adaptation performance effects when compressing a case–base.

Leake [9] states that a case–based reasoner only needs to handle the types of problems that actually occur in practice, while generative systems must account for all possible problems. So, another check might be to look if there are no superfluous cases.

Uncontrolled case base growth can cause serious performance problems as retrieval efficiency degrades and incorrect or inconsistent cases become increasingly difficult to detect. Smyth [17] proposes a competence–based maintenance policy for case–based reasoning systems.

A knowledge light approach to similarity knowledge maintenance is proposed by Patterson, Anand, and Hughes in [13] which updates similarity assessment as changes in task or environment dictate. Two techniques based on introspective feedback are presented. The techniques differ in the quality of competence feedback information they utilize. Both techniques refine the coverage of a case as defined by the original similarity knowledge with an aim to improving the competence of the case–base.

To check retrieval results test sets of queries could be defined by experts. In regular intervals these queries could be asked to the case–based reasoner and the results can be compared. Differences in the result sets may indicate problems in the knowledge containers.

Saving different states of the knowledge containers may be of value in general. Comparing different states allows to make changes visible. The CBR engineer may detect trends to improve maintenance, e.g., when to execute the maintenance steps.

The quality measures are useful only if they are evaluated. Monitoring operators enable permanent control of the quality values and specific indicators lead to the initiation of the restore step. If the quality level decreases to some predefined value the review step suggests specific changes to get back to the desired level of quality.

4.3 Restore

The restore step is described by the tasks *select* and *modify*. This step selects appropriate modify operators and utilizes them to change the contents of the CBR system.

The review step will suggest several possibilities to get back to the quality level desired. Depending on the application domain there may be different selection methods.

In knowledge intensive CBR systems the selection of modify operators could be controlled by the domain theory. In knowledge poor systems a qualified user has to select what to do. For example, in planning domains plans may be deleted automatically using a competence preserving deletion strategy [18]. Just in case the plan was erroneously deleted the generative planner always can reproduce it. In a help–desk domain the CBR engineer should decide whether to delete a case or not [5].

Heister and Wilke [6] describe different operations for the maintenance of the vocabulary knowledge container and repair strategies for the case base. They provide an architecture which ensures that the user is notified if changes on the vocabulary affect other parts of the case–based reasoner.

The modify task consists of two subtasks: *execute operator* and *adjust quality information*. When a modify operator has been selected it may be executed automatically or with the help of a user. Examples for case base modify operators described in [15] are add case, remove case, specialize case, generalize case, adjust case, alter case, cross cases, join cases, combine cases, and abstract cases.

After the modification of the appropriate knowledge containers the quality information must be updated to reflect the change. Also the general maintenance information should be adjusted.

It may be useful to reiterate the review and restore steps several times because of the many dependencies between the elements of the four knowledge containers.

5 Concluding remarks

In summary, we revisited the four step process model of [2] and described major weaknesses regarding the maintenance of case–based reasoning systems. Then, we described our six step process model and a decomposition of the two steps *review* and *restore* into tasks and methods. We related known maintenance methods to the new steps as examples how to accomplish the respective tasks.

We showed that the six step process model overcomes the deficiencies of the traditional cycle. It distinguishes an application and a maintenance phase, handles changes of the system and the environment explicitly, and collects maintenance data throughout the system.

This paper is only a first step on the way to a maintenance methodology. In science a methodology is a set of methods and rules followed [1]. To get a complete set of methods for case–based reasoner maintenance the tasks and methods described in this paper must be examined more closely and developed further in a systematical way. This will be one of our tasks for the near future.

Acknowledgments

The authors thank Thomas Reinartz (DaimlerChrysler, Research and Technology) for the fruitful discussions and his continuous support.

[8]

References

[1] *Webster's New Encyclopedic Dictionary*. Black Dog and Leventhal Publishers Inc., 151 W. 19th Street, New York, New York 10011, revised edition, 1995.

[2] Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.

[3] Ralph Bergmann, Sean Breen, Mehmet Göker, Michel Manago, and Stefan Wess. *Developing Industrial Case–Based Resoning Applications: The INRECA Methodology*. Lecture Notes in Artificial Intelligence, State–of–the–Art–Survey, LNAI 1612. Springer–Verlag, Berlin, 1999.

[4] Maria Angela Ferrario and Barry Smyth. A user-driven, distributed maintenance strategy for large-scale case-based reasoning systems. In Mirjam Minor, editor, *ECAI Workshop Notes*, pages 55–63. Humboldt–Universität zu Berlin, 2000.

[5] Mehmet Göker and Thomas Roth-Berghofer. Development and utilization of a case-based help–desk support system in a corporate environment. In Klaus-Dieter Althoff, Ralph Bergmann, and L. Karl Branting, editors, *Case-Based Reasoning Research and Development*, pages 132–146, Berlin, 1999. Springer-Verlag.

[6] Frank Heister and Wolfgang Wilke. An Architecture for Maintaining Case-Based Reasoning Systems. In Barry Smyth and Pádraigh Cunningham, editors, *Advances in Case-Based Reasoning, Proceedings of the 4th European Workshop on Case-Based Reasoning EWCBR98*, pages 221–232. Springer-Verlag, 1998.

[7] Ioannis Iglezakis and Thomas Roth-Berghofer. A survey regarding the central role of the case base for maintenance in case–based reasoning. In Mirjam Minor, editor, *ECAI Workshop Notes*, pages 22–28. Humboldt–Universität zu Berlin, 2000.

[8] David Leake, editor. *Case–Based Reasoning — Experiences, Lessons, and Future Directions*, Menlo Park, CA, Cambridge, MA, 1996. AAAI Press/MIT Press.

[9] David Leake. CBR in context: The present and the future. In *Case–Based Reasoning: Experiences, Lessons, and Future Directions* [8], pages 3–30.

[10] David B. Leake and David C. Wilson. Remembering why to remember: Performance–guided case–base maintenance. In Enrico Blanzieri and Luigi Portinale, editors, *Advances in Case–Based Reasoning*, pages 161–172. Springer–Verlag, 2000.

[11] Mario Lenz. *Case Retrieval Nets as a Model for Building Flexible Information Systems*. PhD thesis, Mathematisch-Naturwissenschaftliche Fakultät II der Humboldt-Universität zu Berlin, 1999.

[12] Mario Lenz, Brigitte Bartsch-Spörl, Hans-Dieter Burkhard, and Stefan Wess, editors. *Case-Based Reasoning Technology: From Foundations to Applications*. Lecture Notes in Artificial Intelligence. Springer–Verlag, Berlin, 1998.

[13] David Patterson, Sarabjot S. Anand, and John Hughes. A knowledge light approach to similarity maintenance for improving case–base competence. In Mirjam Minor, editor, *ECAI Workshop Notes*, pages 65–78. Humboldt–Universität zu Berlin, 2000.

[14] Thomas Reinartz, Ioannis Iglezakis, and Thomas Roth-Berghofer. On quality measures in case base maintenance. In Enrico Blanzieri and Luigi Portinale, editors, *Advances in Case–Based Reasoning*, pages 247–259. Springer–Verlag, 2000.

[15] Thomas Reinartz, Ioannis Iglezakis, and Thomas Roth-Berghofer. Review and restore for case base maintenance. *Computational Intelligence: Special issue on maintaining case–based reasoning systems*, 17(2):214–234, 2001.

[16] Michael M. Richter. The knowledge contained in similarity measures. Invited Talk at the International Conference on Case–Based Reasoning, 1995. http://wwwagr.informatik.uni-kl.de/∼lsa/CBR/Richtericcbr95remarks.html.

[17] Barry Smyth. Case–base maintenance. In *Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*. Springer-Verlag, 1998.

[18] Barry Smyth and Mark Keane. Remembering to forget: A Competence-Preserving Case Deletion Policy for Case-Based Reasoning Systems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 377–382, 1995.

[19] Barry Smyth and Elizabeth McKenna. Modelling the competence of case-bases. In Barry Smyth and Pádraigh Cunningham, editor, *Advances in Case-Based Reasoning, Proceedings of the 4th European Workshop on Case-Based Reasoning EWCBR98*, pages 208–220. Springer-Verlag, forthcoming 1998.

[20] E. Burton Swanson. The dimensions of maintenance. In *Proceedings of the 2nd International Conference on Software Engineering*, pages 492–497, 1976.

[21] Ian Watson. CBR workshop at ICJAI'99, 1999. http://www.ai-cbr.org/ijcai99/workshop.html.

[22] Stefan Wess. *Fallbasiertes Problemlösen in wissensbasierten Systemen zur Entscheidungsunterstützung und Diagnostik*. Dissertation, Universität Kaiserslautern, 1995.

[23] Wolfgang Wilke. *Knowledge Management for Intelligent Sales Support in Electronic Commerce*. Dissertation, Universität Kaiserslautern, 1999.

[24] Wolfgang Wilke, Ivo Vollrath, and Ralph Bergmann. Using knowledge containers to model a framework for learning adaptation knowledge. In Dietrich Wettschereck and David W. Aha, editors, *European Conference on Machine Learning. MLNet Workshop Notes - Case-Based Learning: Beyond Classification of Feature Vectors*, pages 68–75, Naval Research Laboratory, Washington, D. C., USA, 1997. Navy Center for Applied Reasearch in Artificial Intelligence.