

Maintenance Memories: Beyond Concepts and Techniques for Case Base Maintenance

Ioannis Iglezakis¹, Thomas Reinartz¹, and Thomas Roth-Berghofer²

¹ DaimlerChrysler AG, Research and Technology, RIC/AM,
P.O. Box 2360, 89013 Ulm, Germany

`ioannis.iglezakis@daimlerchrysler.com`

`thomas.reinartz@daimlerchrysler.com`

² German Research Center for Artificial Intelligence DFKI GmbH,
Erwin-Schrödinger-Straße 57, 67655 Kaiserslautern, Germany

`thomas.roth-berghofer@dfki.uni-kl.de`

Abstract. Maintenance of Case-Based Reasoning (CBR) systems became an important area since applications of CBR technologies were established in different real-world domains. Maintenance issues cover all aspects that help to keep a running CBR system in a usable state of high quality. Concepts and techniques that were developed for maintenance of CBR systems range from methodologies and frameworks that particularly define phases, steps, and tasks necessary to integrate maintenance into the CBR process up to specific programs that enable CBR engineers to carry out the maintenance activities. In this paper, we exemplify this range of research on maintenance of CBR systems by brief characterizations of the SIAM methodology, the MAMA maintenance manual, and the MASH maintenance shell. The overall goal of this paper is then to conclude areas for further research in maintenance for CBR systems from the experience of the work on SIAM, MAMA, MASH, and related approaches.

1 Introduction

Case-Based Reasoning (CBR) has become well-known as a viable technology for problem solving, finding similar objects, or classifying items with unknown class labels. CBR became an accepted technology, which is now applied in many different application domains. However, most research covered only aspects of defining vocabularies for knowledge representation, building up cases and case bases, developing sophisticated organizational structures of case bases, and implementing intelligent similarity measures and retrieval mechanisms. Most of these efforts were about usage of CBR, but did not deal with methods for keeping a CBR system up and running over longer periods of time.

Driven by practical applications of CBR in real-world domains such aspects of *maintenance* arose at the end of the last century. The CBR community and system providers became aware of issues such as dealing with changes in the environment of the application domain, handling quality issues of the CBR system, optimizing retrieval performance over time, or reacting on new customer requirements.

Since this awareness, a rather small number of people in the CBR community started to think about maintenance issues, and to come up with solution approaches for different maintenance purposes. Work in the area of maintenance for CBR systems covered both the development of methodologies and maintenance task models as well as the specification of guidelines for how to perform maintenance and concrete algorithms and systems to execute programs that deal with maintenance tasks.

The overall goal of this paper is to recapitulate research on CBR maintenance and to draw conclusions that point to limitations of existing approaches and propose directions for future work that go beyond the current state of maintenance research. In particular, we exemplify the discussion on different work on case base maintenance by remembering research on the SIAM methodology, the MAMA maintenance manual, and the MASH maintenance shell.

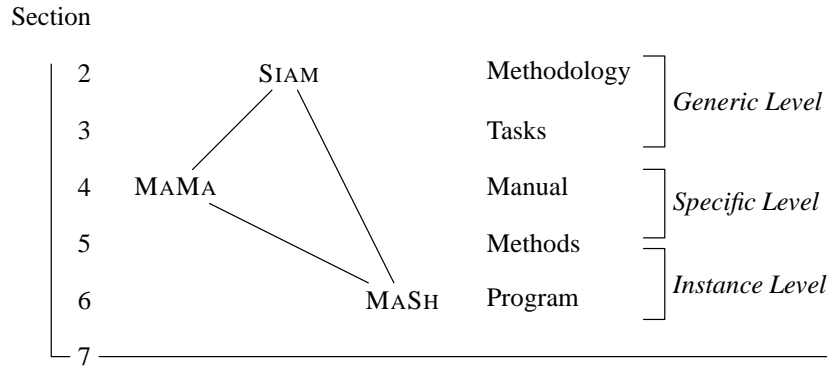


Fig. 1. Maintenance Research at Different Levels of Abstraction

Figure 1 shows an overview of the structure of this paper along the different levels of abstraction of research on maintenance for CBR systems and their relation to each other. In the following section, we describe SIAM, a methodology that covers all phases of CBR starting with the setup and initialization of a CBR system up to its application and maintenance. SIAM is at the most generic level of abstraction and is general enough to hold for any CBR application.

The third section is devoted to a more specific description of maintenance tasks that define which generic activities belong to the different phases of the SIAM methodology. In this paper, we focus on task descriptions for the maintenance phase of SIAM. Within the fourth section, we exemplify contents of the maintenance manual MAMA. This manual is at the first specific level of abstraction and includes more concrete guidelines how to perform maintenance in detailed situations. Hence, MAMA is a specialization of (the maintenance phase of) the SIAM methodology.

The most specific part of this paper at the instance level of abstraction elaborates methods and programs that enable maintenance engineers to actually run executable code to perform the maintenance tasks. The fifth section describes methods of case

properties and performance measures that define quality measures for case base maintenance as well as operators that modify cases to perform case base maintenance on concrete case bases. Thereafter, the sixth section is on the maintenance shell MASH, which implements the described methods, and its evaluation within a few example domains. MASH achieves both, it specializes the methodology SIAM as well as implements techniques for the maintenance manual MAMA.

The seventh section summarizes lessons learned from the work on SIAM, MAMA, and MASH. Thereby, we discuss several limitations at all three levels of abstraction and suggest issues for future work that go beyond current efforts in the area of maintenance for CBR systems.

2 Maintenance Methodologies

Methodologies, at the top most level of abstraction, define frameworks that categorize different aspects of a discipline in order to structure issues related to this discipline. Moreover, methodologies often cover process models that describe the general workflow and the various tasks for some field. For maintenance of CBR systems, the following subsections describe such categorizations as well as SIAM as an example for a well-structured process model for CBR in general and maintenance of CBR systems in particular.

2.1 Types of Maintenance

Knowledge maintenance of CBR systems [17] partly builds upon principles of software maintenance, which aims at eliminating errors and adapting software to changes of user requirements regarding functionality and performance. Swanson [21] distinguished *corrective*, *adaptive*, and *perfective* maintenance.

Corrective maintenance deals with processing failure, which is due to incorrect computations. Other failures are performance failures (i.e., performance criteria are not met) and implementation failures.

Adaptive maintenance is needed whenever the environment of a program changes, typically leading to corrective maintenance. Anticipating environment changes to avoid such failures leads to adaptive maintenance.

Perfective maintenance is performed to eliminate processing inefficiencies, to enhance performance, or to improve maintainability. This type of maintenance is directed to keep a program up and running at less expense, or to better serve the needs of its users.

These three types of maintenance can be grouped as *function preserving maintenance* in contrast to *function enhancing* and *supporting maintenance* [10]. *Function enhancing maintenance* aims at the implementation of further functionality into the system. User training, help in using the system, and planning of maintenance activities are examples of *supporting maintenance*. Our own work concentrated on function preserving maintenance. Function enhancement is, in our view, similar to development tasks. For this and supporting maintenance we refer to the INRECA methodology [2].

2.2 The SIAM Methodology

After considering the different types of knowledge maintenance, we looked at CBR process models as the starting point for the development of a maintenance methodology. Early on, CBR has been decomposed into sub processes. Riesbeck and Bain [15] illustrated the basic process of CBR in a flowchart, and Kolodner [8] described CBR as a process of remember and adapt or remember and compare. The most influential model of CBR is the 4RE process according to Aamodt and Plaza [1] with its four steps *retrieve*, *reuse*, *revise*, and *retain*. This four step process was the foundation for the formulation of the SIAM methodology [16].

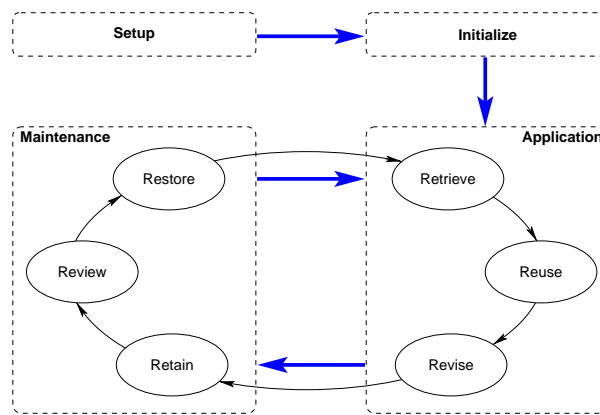


Fig. 2. The SIAM methodology and the six steps process model

SIAM, described as an extension or completion of the INRECA framework [2], is a methodology for developing and maintaining CBR solutions. Figure 2 shows an overview of the SIAM process model comprising the four phases *setup*, *initialization*, *application*, and *maintenance*.

Setup addresses all issues that a CBR project has to deal with in its early stages such as setting up the project's goals and designing the CBR system, always staying focused on the customer's aims. In the initialization phase, the knowledge containers are filled as soon as the CBR system has been implemented or configured. Then, the CBR system is ready for use and enters the phases application and maintenance in turn.

The application phase comprises the three steps *retrieve*, *reuse*, and *revise* of the original four steps process model [1]. Those three steps do not change the knowledge of the CBR system. As soon as knowledge is to be learned, e.g., by storing revised cases in the *retain* step, the maintenance phase begins. This maintenance phase contains the steps *retain*, *review*, and *restore*. All in all, we thus proposed a six steps process model for handling application and maintenance of CBR systems [13].

3 Maintenance Tasks

The steps *review* and *restore* were introduced to handle the maintenance control loop [17]. They are further divided into tasks at the next lower level of abstraction. In terms of the control loop metaphor, the review step observes the current state of a given CBR system, measures its quality, and invokes the restore step if necessary. In the review step, we distinguish the tasks *measure* and *monitor*. The restore step, responsible for bringing the CBR system back to a desired quality level, actually changes the CBR system. Its tasks *suggest*, *select*, and *modify* are executed as soon as the review step indicates the necessity. In the following, we describe the five tasks at a generic level.

Measure. Before any repair can take place one has to determine what to repair, i.e., some kind of quality assessment has to take place. In our framework, the measure task computes an analysis of the current quality level with respect to the knowledge of the case-based reasoner, thus providing a measurement of the current quality level as an input to the monitor task.

Monitor. The outcomes of the measure task are fed into the monitor task for further evaluation. In interactive scenarios, the measurements can be visualized for discussion among the maintenance engineers and domain experts. In automated scenarios, measurements are analyzed and compared to past quality measurements as well as to specifications of the setup and initialization phases. The main goal of the monitor task is to decide *if* there is a decline in quality and if it is necessary to react to such a decline. If the respective maintenance engineer or program comes to the conclusion that maintenance is imminent, a notification must be issued.

Suggest. The suggest task compiles a list of alternative possible repair operations on the basis of the measurements, taken during execution of the measure task, and on the basis of the monitor task's outcomes. Each of the repair operations should be accompanied by its (estimated) execution costs.

Select. From the list of applicable repair operations, the select task chooses the appropriate ones. Therefore, the operations are ranked by, e.g., estimated execution costs or the number of side effects of each operation. In knowledge intensive scenarios, where actually changing the CBR system would be too costly, it may be even appropriate to simulate the execution of different promising sets of proposed operations and then to check the quality level of the different generated CBR systems, in order to select the best set of repair operations. The main goal of the select task is to decide *what* to do.

Modify. Finally, the selected repair operations are executed in the modify task, thus updating the knowledge containers accordingly. The outcome of the operations should be checked again by returning to the measure task. An inner maintenance loop of review and restore could be used to increase the level of quality until a desired threshold is reached.

4 Maintenance Manuals

The maintenance manual MAMA is located at the specific level. At this level of description, the context for such a maintenance manual is fixed with respect to the type of

CBR, type of CBR application, type of CBR tool, and affected knowledge containers. The user of such a manual is guided from setting up maintenance and initializing all necessary parameters up to performing appropriate activities in certain situations.

Each maintenance task within the maintenance manual is an atomic unit of description of our process model and consists of several components: *inputs*, *activities*, *methods*, *resources*, and *outputs* [16].

The *inputs* describe whatever is necessary before the respective task can be carried out, e.g., outputs from a previously executed task or required data in form of reports, evaluations, or management decisions. A series of *activities* describes how the task could be accomplished using the task's *methods*. Such methods need not only be technical ones but could also be managerial or organizational techniques. Another component states which *resources* are required to perform the task. We distinguish between two types of resources, namely human resources and (software) tools. Finally, the *outputs* element of a task specifies the expected results of carrying out this task.

The maintenance manual addresses the maintenance engineer and the CBR system administrator, not the CBR system user [17]. The manual users are mostly interested in *when* to maintain the respective CBR system and in *what* to do. Event-Condition-Action (ECA) rules [14] are used to structure the maintenance manual. Events from organizational processes (e.g., timer-based events) trigger the review step which, in turn, activates restore operations. An example maintenance manual for EMPOLIS ORENJE used in case-based decision support can be found in [17].

5 Maintenance Methods

Now, we turn to the next lower level of abstraction in the hierarchy of concepts and techniques for case base maintenance. This section on maintenance methods describes specific techniques to carry out the various activities defined in MAMA, and to work on the tasks specified within the SIAM methodology, respectively.

5.1 Performance Measures

First, we describe concrete performance measures that implement quality measures for case base maintenance and correspond to typical customer requirements that use a CBR system in practice [7,6]. Since customer requirements vary for different types of applications, we restrict our view on customer requirements in classification domains.

Coverage. The first customer requirement denotes that customers want an answer to their problems. The corresponding performance measure is coverage P_V . We assume, a case base *covers* a customer's query if there exists a case within the case base that is at least as similar to the query as a pre-defined similarity threshold τ . The overall *coverage* of a case base in relation to a set of queries is the total number of covered queries divided by the total number of queries in the query set.

Positive Coverage. An extension of coverage is positive coverage P_V^+ , which additionally considers the correctness of the retrieved answer. A case base *correctly covers* a query if there exists a case within the case base that covers the query and the

solution of this case solves the problem of this query. The overall *positive coverage* of a case base in relation to a set of queries is the total number of correctly covered queries divided by the total number of queries in the query set.

Accuracy. The third performance measure, accuracy P_A^+ , also considers correctness of answers to customer's queries. But rather than requiring a minimum similarity τ as positive coverage, accuracy takes into account the correctness of the actual solution of the retrieved most similar case given a query. We say, a case within the case base *classifies* a query, if the problem description of this case is most similar to the query. Similarly, a case *correctly classifies* a query, if the case classifies the query and the solution of the case actually solves the query. The *accuracy* of a case base in relation to a set of queries is the number of correctly classified queries divided by the total number of queries in the query set.

Confidence. The next performance measure is *confidence* P_C , which is the average similarity of all classified queries by a case base in relation to a set of queries. Confidence is a measure that corresponds to the customer requirement that expects for each received solution to a query a high probability of correctness.

Positive Confidence. As for coverage and positive coverage, we also extend confidence to positive confidence P_C^+ , which is the average similarity of all correctly classified queries by a case base in relation to a set of queries.

Storage Space. Finally, the last performance measure takes into account that retrieval speed is related to costs and hence, customers want quick answers to their problems. In turn, retrieval time is related to storage space. Hence, we define the performance measure for *storage space* P_T , which is the total number of cases in the case base.

Although the performance measures reflect the customer requirements well, there are two fundamental drawbacks. First, the performance measures provide no information how to maintain a CBR system. Thus, we cannot infer the maintenance operations necessary to restore the quality of a CBR system from values of performance measures. Second, we must apply a CBR system before we can compute the performance measures. For both reasons, we have to find a way to measure some criteria that reflect these performance measures and that are computable before really using the system in practice. In addition, such criteria should provide hints for specific maintenance operations to restore a CBR system if desired.

5.2 Case Properties

We define the respective criteria to overcome the previously mentioned drawbacks with the use of case properties, which indicate conflicts between cases before the CBR system is in real use. Later on in the next subsection, we also define modify operators to eliminate the indicated conflicts. Hence, the case properties and their relation to the modify operators allow direct clues for concrete maintenance operations.

In this paper, we shortly recapitulate four basic case properties (see also [13] and [6] for formal definitions):

Consistent. A case $c = (p, s)$ with problem description p and solution s is *consistent*, if there does not exist any other case $c' = (p', s')$ for which the problem description p' is the same or more general as p and the solution s' is different to s .

Unique. A case $c = (p, s)$ with problem description p and solution s is *unique*, if there does not exist any other case $c' = (p', s')$ for which the problem description p' is the same as p and the solution s' is the same as s .

Minimal. A case $c = (p, s)$ with problem description p and solution s is *minimal*, if there does not exist any other case $c' = (p', s')$ for which the problem description p' is the same or more general as p and the solution s' is the same as s .

Incoherent_Δ. A case $c = (p, s)$ with problem description p and solution s is *incoherent_Δ*, if there does not exist any other case $c' = (p', s')$ for which the attribute values of problem description p' overlap with the attribute values of problem description p except for a small specific number ($Δ$) of attribute values and the solution s' is the same as s .

5.3 Modify Operators

As mentioned previously, modify operators allow to restore the quality of the case base. Here, we shortly describe five modify operators that work on single cases and two modify operators that work on two cases. Again, we refer to [13] and [6] for complete formal definitions:

The *Remove Case* modify operator deletes a specific case from the case base C .

The *Specialize Case* modify operator adds an attribute value to a specific case.

The *Generalize Case* modify operator deletes an attribute value of a specific case.

The *Adjust Case* modify operator changes an attribute value of a specific case.

The *Alter Case* modify operator deletes an attribute value of a specific case and subsequently adds a different attribute and its value to this case.

The *Cross Cases* modify operator reduces two cases into one case by building the intersection of the two problem components. The two cases must be either not incoherent_Δ or the one case is not minimal in comparison to the other case.

The *Join Cases* modify operator reduces two cases into one case by building the union of the two problem components. The two cases must be either not incoherent_Δ in a way that the differences do not share an attribute or the one case is not minimal in comparison to the other case.

6 Maintenance Programs

At the most specific level of maintenance, we have concrete programs that enable the maintenance personnel to accomplish the activities described in the maintenance manual MAMA, or to perform the maintenance tasks defined in the methodology SIAM.

By now, several programs for maintenance of CBR systems have been proposed. For example, Smyth and McKenna [19] developed a tool for competence-guided authoring and visualization of cases. Another system helps to maintain the case base through continuous support for case authoring and design consistency in the domain of aerospace design [22]. McSherry [12] described CaseMaker-2, which also supports authoring of cases by indicating uncovered spaces in the case base that show potentials for adding new cases interactively by the maintenance personnel. Other work on maintenance programs includes the maintenance management system Dr. orange [11] that

implements the review step of the SIAM methodology and supports different types of measures such as case properties that are monitored for interactive maintenance as well as the BASTIAN (case BAsed SysTem In cLAssificationN) platform, which implements an additional component for automatic maintenance using rough sets [18].

6.1 MASH—The Maintenance Shell

In this paper, we focus on a more detailed description and evaluation of the maintenance shell MASH, which is a complete implementation of the maintenance phase of SIAM at the instance level. MASH implements the review and restore steps and their corresponding tasks measure, monitor, suggest, select, and modify. MASH supports automatic as well as interactive execution of these tasks.

The complete maintenance shell realizes the tasks as components with an additional preceding component that reads a data stream and with a following component that implements a simple CBR algorithm for simulation purposes. The current implementation of MASH works on the case base, requires no domain knowledge, and focuses on classification domains with no adaptation. The different components of the implementation use the *factory*, *flyweight*, and *singleton* design patterns as described by Gamma et al. [4]. Thereby, we can easily extend the actual implementation. For example, the factory pattern allows exchanging different component objects without changing the underlying code structure.

6.2 Evaluation of MASH

To exemplify that maintenance applications like MASH actually work in practice, we present the results of an evaluation. The first purpose of this evaluation is to show effects on the performance measures when modify operators are applied in a 10-fold cross validation to a case base with the help of the case properties. The second purpose is to show the robustness of the case properties and modify operators in comparison to the corresponding unchanged case base when the size of the case base varies, but the test case base remains the same.

Experimental Design. The evaluation uses the crx case base for credit approval from the UCI machine learning repository [3]. This case base has 690 cases that consist of 15 attributes (9 nominal and 6 numeric) with missing values and two classes. We performed a 10-fold cross validation with a simple 1-nearest-neighbor algorithm with 5%, 10%, 15%, ..., 100% of the original training set as varying training case bases. The used case properties consistency, minimality, uniqueness, incoherence₁, and incoherence₂ detected conflicts that we maintained with a modify operator to record the performance measures and to compare these values with the results of the performance measures for the corresponding unchanged case base.

Evaluation Results. Table 1 shows the results of the different modify operators and the performance measures in comparison to the corresponding unchanged case base when 100% of the original training set is used as the training case base.

Table 1. Results on the whole crx case base for the performance measures and the modify operators using the case properties consistency, minimality, uniqueness, incoherence₁, and incoherence₂.

	P_V	P_V^+	P_A^+	P_C	P_C^+	P_T
None	95.07	92.46	89.13	86.82	77.18	621.0
Adjust Case	94.64	91.74	88.84	86.68	76.88	621.0
Alter Case	95.07	92.46	89.42	86.80	77.44	621.0
Cross Cases	95.94	93.77	87.68	87.35	76.34	605.8
Remove Case	93.62	90.14	88.12	85.09	74.77	409.9
Generalize Case	95.51	93.62	90.58	87.44	79.00	621.0
Join Cases	95.07	92.32	88.99	86.81	77.05	620.2
Specialize Case	95.07	92.46	89.57	86.78	77.57	621.0

The overall best results for the performance measures coverage P_V , positive coverage P_V^+ , accuracy P_A^+ , confidence P_C , and positive confidence P_C^+ shows the modify operator generalize case, followed by specialize case. In contrast, the overall poorest results for these performance measures shows the remove case modify operator. For the performance measure storage space P_T , the remove case modify operator shows the best result.











































Table 2 displays in a qualitative way the results for the performance measures over different training case base sizes. This robustness tells us something about the effects on the performance measures applying a modify operator on different sizes of training case bases. Each of the ticks represents the value of a performance measure in a modified case base minus the corresponding value of the unchanged case base. Every first tick is computed on 5% of the training case base, while the last tick is computed on 100% of the training case base. Hence, the values of the last ticks correspond to the values of table 1. For example, the coverage for the case base that is changed with the adjust case modify operator is 94.64% and the corresponding coverage for the unchanged case base is 95.07%. The difference is -0.43% , which is the amplitude of the last tick in the corresponding diagram of table 2.

For the performance measures coverage, positive coverage, accuracy, and confidence, the results of the modify operators cross cases, generalize case, and remove case are robust over the changing training case base size. For positive confidence, the results of the modify operators generalize case and remove case are robust over the changing training case base size. Finally, for storage space, the results of cross cases and remove cases are robust. These results for robustness help us to decide which modify operator we can use to improve the quality of the resulting case base. Hence, for maintenance of case bases like crx we advice to use the generalize modify operator.

7 Further Maintenance Research Issues

In this paper, we described various example research contributions to maintenance for CBR systems at different levels of abstraction. At the generic level, we briefly summarized the SIAM methodology and its maintenance task descriptions. At the specific

Table 2. Qualitative results on $\{5\%, 10\%, 15\%, \dots, 100\%\}$ of the crx case base for the subtracted performance measures and the modify operators using the case properties consistency, minimality, uniqueness, incoherence₁, and incoherence₂.

	P_V	P_V^+	P_A^+	P_C	P_C^+	P_T
Adjust Case						
Alter Case						
Cross Cases						
Remove Case						
Generalize Case						
Join Cases						
Specialize Case						

level, we outlined the maintenance manual MAMA and exemplified maintenance methods based on case properties and modify operators. And finally, at the instance level, we introduced several maintenance programs, especially the maintenance shell MASH. The evaluation of MASH showed that our well-structured approach along all levels of abstraction yields satisfying results for maintenance in practice.

In this section, we put together conclusions from lessons learned in about five years maintenance research and some initial practical experience that we made in applying the methodology and techniques briefly presented in this paper. These conclusions result in several open issues and in suggestions for future research in case base maintenance and beyond. These topics can form the basis for further research in the area of maintenance for CBR systems.

7.1 Beyond Current Limitations

The first subsection of future work covers concrete aspects that aim to overcome specific limitations of the presented approaches.

More Experiments in real Real-World Settings. We mostly tested our approaches in domains of the UCI machine learning repository. Furthermore, we used MASH within one industrial project on case-based help-desk support. Although results in this project were very promising, we encourage more experiments in more industrial settings to get more experience in "real" practical applications.

Meta Information. In several publications, we mentioned an additional case component, called the meta information q . For example, this meta information includes coun-

ters on usage patterns of cases. The basic idea for meta information q that we proposed within the framework of SIAM, MAMA, and MASH still needs more work to specifically define the concrete measures with use of this type of meta information and their applications for maintenance.

Alternative Quality and Performance Measures. In a similar vein, we proposed several quality and performance measures for case base maintenance. Again, we can think of alternative measures that take into account completely novel aspects such as the meta information q . Similarly, there is still potential for the development of more techniques to control the quality of a CBR system and to monitor when a CBR system needs maintenance. For example, time series prediction techniques may take series of quality measure's values and predict future values. Thereby, maintenance becomes possible in a preventive manner, even before the CBR system does not fulfill any quality criteria any longer.

Maintenance Advice. Another potential of improvement in case base maintenance is the definition of heuristics that suggest in which specific situations which concrete maintenance operators are best for applications. Initial work along such heuristics showed that proposals at the instance level often depend on the characteristics of the application domain and that it is hard to explicitly identify which characteristic is responsible for which effect. Consequently, we believe that more work is necessary here.

Case-Base Versioning. Another idea in the area of case base maintenance is the concept of keeping a trace of case base versions. One possibility of this kind of case base versioning is to keep copies of different states of the case base. An alternative way is to store only protocols of changes between different versions of the case base. In any case, versioning allows to trigger maintenance by reasoning on differences between complete case bases rather than only considering single cases or pairs of cases. Again, we believe that this aspect of diachronic approaches [9] possibly opens new opportunities for case base maintenance.

Beyond Function preserving Maintenance. Finally, we conclude that most of the research on case base maintenance is only function preserving and does not yet deal with function enhancing maintenance.

7.2 Outside Case-Base Maintenance

This subsection on issues for future maintenance research emphasizes that current research mostly deals with maintenance of the case base, mainly neglecting the three other containers vocabulary, similarity measures, and adaptation knowledge.

Vocabulary Maintenance. Maintenance that takes into account the vocabulary container relates to representation issues in CBR. For example, vocabulary maintenance should detect missing attributes, redundant or unnecessary attributes, or potentials for generalizations of different attributes. Vocabulary maintenance must be able to perceive and handle problems with values of attributes' domains [5]. Changes of the vocabulary often result in the necessity for changes of all the other containers.

Similarity Measures Maintenance. Maintenance on similarity measures is most likely to analyze the retrieval behavior of a CBR system. Automatic learning of attribute weights is an example of such maintenance, and most existing research on maintenance related to similarity measures is within this area (e.g., [20]). A potential drawback of this type of maintenance is the sort of locality that changes of similarities possibly mean. If for some cases an adapted similarity measure sorts retrieval results in a more meaningful sense, for other cases the same adaptation might result in the contrary.

Adaptation Knowledge Maintenance. Maintenance on adaptation knowledge is the most difficult issue from our perspective. Although initial work towards maintenance of adaptation knowledge exists, this type of maintenance is probably highly application dependent as the usage of adaptation knowledge itself is. Hence, we expect that solutions for adaptation knowledge maintenance are not generally applicable, and that results in this area will only be at the specific or instance level of abstraction.

Relations between Knowledge Container Maintenance. Another idea for future work is the analysis of relations between the different knowledge containers and maintenance operations on them. For example, if we maintain the case base, it is not explicitly necessary to change one of the other knowledge containers since there is no explicit relation between the case base and vocabulary, similarity measures, and adaptation knowledge. However, changing the case base might lead to a different retrieval behavior of a CBR system, and hence it also might then make sense to think about maintenance of the similarity measures container as well.

7.3 Towards New Maintenance-related Areas

The third list of topics for future work is more on maintenance-related issues that generally deal with tasks that enable or improve maintenance but that do not directly contribute to specific maintenance solutions.

Environment Changes. As mentioned before, changes in the environment of an application outside the CBR system in use often result in the need for maintenance. However, it is an open question how to recognize changes of the environment and which type of changes are possible at all. Whereas the automatic detection of such changes is perhaps not possible, it is reasonable to think about strategies that guide maintenance for specific types of changes. ECA rules as mentioned in section 4 are possibly one potential mechanism to handle environmental changes.

Domain Knowledge. Up to now, maintenance research does not utilize domain knowledge, except for the knowledge represented in one of the knowledge containers of the CBR system. If we think of domain dependent rules, that describe relationships between attribute values across different attributes, for example, we can also imagine using these rules not only for query completion but also for maintenance operations.

Maintenance Explanations. For all changes that maintenance performs on a CBR system, there is a specific reason. Hence, it is helpful for the maintenance personnel, to understand why a maintenance program such as MASH proposes some maintenance operations for execution. Consequently, an explanation component of a maintenance system is a reasonable extension of current maintenance capabilities to increase acceptance of automatic and computer-assisted maintenance for CBR systems.

Maintenance Tutoring System. When a component for maintenance is available, it is possibly not straight-forward when and how to use it. A maintenance tutoring system, along with maintenance examples or e-learning modules, is consequently again a helpful extension for a maintenance system that aids maintenance personnel in their daily job. Such tutoring also leads to consistent maintenance behavior among the different maintenance engineers, and can be classified as supporting maintenance.

Maintenance Support by Maintenance Programs. A similar idea proposes to utilize maintenance programs only to support maintenance personnel that performs maintenance manually. For example, if a maintenance engineer decides to remove a case since it is in conflict with another case, MASH possibly evaluates this specific change in advance and suggests an alternative operation that does not only resolve this conflict by removing this case but eliminates more conflicts by modifying a different case.

Manual vs. Automatic Maintenance. The last idea to use maintenance programs to support manual maintenance opens the discussion on the relation between human-centered and automatic maintenance. It is interesting to analyze the limitations of both approaches and to develop processes that enable best mutual benefits from both approaches. In this paper, MAMA is an example for the human-centered approach whereas MASH is an automatic instrument.

Meta Maintenance. Finally, while seeking the memories of maintenance research, we recalled the idea of meta maintenance (e.g., [9]). This meta maintenance means that it is also important to keep track of maintenance operations, of the resulting changing quality of the CBR system, and whether automatic maintenance operations are accepted by the maintenance personnel. These considerations lead to an analysis of the maintenance itself. If we detect that maintenance does not lead to high quality CBR over time or that maintenance operations suggested by the maintenance system are seldomly accepted by the personnel, it is necessary to maintain the maintenance program itself.

References

1. Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.
2. Ralph Bergmann, Sean Breen, Mehmet Göker, Michel Manago, and Stefan Wess. *Developing Industrial Case-Based Reasoning Applications: The INRECA Methodology*. Lecture Notes in Artificial Intelligence, State-of-the-Art-Survey, LNAI 1612. Springer-Verlag, Berlin, 1999.

3. Catherine L. Blake and Christopher J. Merz. UCI repository of machine learning databases, 1998.
4. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1997.
5. Frank Heister and Wolfgang Wilke. An Architecture for Maintaining Case-Based Reasoning Systems. In Barry Smyth and Pádraigh Cunningham, editors, *Proceedings of the Fourth European Workshop on Case-Based Reasoning, EWCBR 98, Dublin, Ireland*, pages 221–232, Berlin, 1998. Springer-Verlag.
6. Ioannis Iglezakis. *Case-Base Maintenance of Case-Based Reasoning Systems in Classification Domains: Methods, Implementation, and Evaluation*. submitted PhD thesis, University of Würzburg, 2004.
7. Ioannis Iglezakis and Thomas Reinartz. Relations between customer requirements, performance measures, and general case properties for case base maintenance. In Susan Craw and Alun Preece, editors, *Proceedings of the 6th European Conference on Case-Based Reasoning (ECCBR)*, pages 159–173. Springer-Verlag, 2002.
8. Janet Kolodner. *Case-Based Reasoning*. Morgan Kaufmann Publishers, Inc., 1993.
9. David B. Leake and David C. Wilson. Categorizing Case-Base Maintenance: Dimensions and Directions. In *Proceedings of the 4th European Workshop on Case-Based Reasoning, EWCBR98*, pages 196–207, Berlin, 1998. Springer-Verlag.
10. Franz Lehner. Ergebnisse einer Untersuchung zur Wartung von wissensbasierten Systemen. *Information Management*, 2:38–47, 1994.
11. Rainer Maximini. Basesystem for maintenance of a case-based reasoning system. Diploma thesis, University of Kaiserslautern, 2001.
12. David McSherry. Intelligent case-authoring support in casemaker-2. *Computational Intelligence: special issue on maintaining CBR systems*, 17(2):331–345, 2001.
13. 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.
14. Joachim Reinert and Norbert Ritter. Applying ECA-rules in DB-based design environments. In *Tagungsband CAD'98 "Tele-CAD—Produktentwicklung in Netzen"*, pages 188–201. Informatik Xpress 9, 1998.
15. C. Riesbeck and W. Bain. A methodology for implementing case-based reasoning systems. Technical report, Lockheed, 1987.
16. Thomas Roth-Berghofer and Thomas Reinartz. MaMa: A maintenance manual for case-based reasoning systems. In David W. Aha and Ian Watson, editors, *Proceedings of the Fourth International Conference on Case-Based Reasoning, ICCBR 2001, Vancouver, Canada*, pages 452–466, Berlin, 2001. Springer-Verlag.
17. Thomas R. Roth-Berghofer. *Knowledge Maintenance of Case-Based Reasoning Systems—The SIAM Methodology*, volume 262 of *Dissertationen zur Künstlichen Intelligenz*. Akademische Verlagsgesellschaft Aka GmbH / IOS Press, Berlin, Germany, 2003.
18. Maria Salamó, Elisabet Golobardes, David Vernet, and Mireya Nieto. Weighting methods for a case-based classifier system. In *Proceedings of the IEEE Learning'00*, 2000.
19. Barry Smyth and Elizabeth McKenna. Competence models and the maintenance problem. *Computational Intelligence: special issue on maintaining CBR systems*, 17(2):235–249, 2001.
20. Armin Stahl. *Learning of Knowledge-Intensive Similarity Measures in Case-Based Reasoning*. PhD thesis, University of Kaiserslautern, 2003.
21. E. Burton Swanson. The dimensions of maintenance. In *Proceedings of the 2nd International Conference on Software Engineering*, pages 492–497, 1976.
22. David C. Wilson. *Case-Base Maintenance: The Husbandry of Experience*. PhD thesis, Indiana University, 2001.