Explanation-Aware Maintenance of Distributed Case-Based Reasoning Systems Thesis Proposal

Pascal Reuss[1] **and Klaus-Dieter Althoff**[2]

[1]University of Hildesheim, Institute of Computer Science
[2]Competence Center Case-Based Reasoning, German Research Center for Artificial Intelligence, Kaiserslautern
reusspa@uni-hildesheim.de, klaus-dieter.althoff@dfki.de

Abstract

This paper is a thesis proposal and describes the idea of an integrated, explanation-aware maintenance approach for distributed case-based reasoning systems. We describe the related work and the foundations our own research will base on and the derived research goals for the dissertation. We describe in detail the integration of several existing approaches with our ideas to develop a methodology and maintenance strategy for distributed case-based reasoning systems.

1 Introduction

Developing and implementing knowledge-based systems, especially case-based reasoning (CBR) systems, has been explored for several years. Today we have methodologies to define and implement knowledge-based or especially case-based reasoning systems. For our research we will first focus on CBR systems and later trying to generalize the approach. For maintaining a single case-based reasoning system there are also several approaches that deal with maintaining the case base, the similarity or the adaptation knowledge. In general all the knowledge sources belonging to a knowledge-based system have to be considered, too. A knowledge source in this context is a software agent with access to a CBR system. These knowledge sources have dependencies between each other that have to be taken into account for a maintenace approach for distributed CBR systems. This thesis proposal describes the idea of an integrated, semi-automatic maintenance approach for distributed CBR systems with explanation capabilities on maintenance.

In Section 2 related work and foundations are described which are underlying our ideas. Section 3 presents the research goals based on these ideas. Section 4 further describes what we mean by explanation-aware maintenance. At last, Section 5 summarizes the paper and gives a short outlook on the next steps.

1.1 Example Domain and Application

In this subsection we describe the application *docQuery* which is settled in the travel medicine domain. The travel medicine domain and the *docQuery* application are used below to illustrate the approach presented in this paper. This application is an mulit-agent-system that uses several knowledge sources to find a solution to a user-given query. The query can contain information like travel destination, planned arrival, age, activities and chronical diseases. Based on this information, the necessary knowledge source, in this case software agents with CBR systems, are identified and requested. The *docQuery* application contains seven different CBR systems that represents several sub-domains of the travel medicine domain like regions, diseases, medication or activities [Bach, 2012][Reuss, 2012].

2 Related Work and Research Basics

This section contains related work that is used as foundations for the ideas in this paper. In his PhD thesis Carsten Tautz developed a methodology for experience management systems called DISER [Tautz, 2000]. This methodology does not consider maintenance in detail, but the methodology can be used as a basis for an integrated maintenance approach. Based on DISER, Markus Nick developed DILLEBIS [Nick, 2005]. This methodology focuses on maintenance that is based on user feedback to identify necessary maintenance actions. Both methodologies are not specifically designed for CBR sytems like INRECA [Bergmann *et al.*, 2003][Althoff and Weis, 1996] is. The latter is focusing on developing CBR applications, but does not explicitly consider maintenance tasks. A methodology for CBR systems originating from the INRECA context and focusing on the maintenance task is Roth-Berghofer's SIAM methodology [Roth-Berghofer, 2003]. This methodology extends the CBR cycle from Aamodt und Plaza [Aamodt and Plaza, 1994] with two steps called *Review* and *Restore*. These steps contain tasks for evaluating and maintaining a CBR system. However, the methodology does not consider distributed CBR systems.

The SEASALT architecture [Bach, 2012][Reichle *et al.*, 2011] provides a general framework for developing distributed knowledge-based systems. The architecture is domain-independent and modular and allows the creation of customized systems. The so-called Knowledge Line is responsible for managing the different knowledge sources. A software agent coordinates the queries and answers to and from the knowledge sources [Reichle-Schmehl, 2008][Bach *et al.*, 2008]. An approach for automating the selection of the needed knowledge sources using CBR was presented by Reuss [Reuss, 2012]. Our future research will base on the SEASALT architecture as a method for developing distributed knowledge-based systems. Additionally we will focus on the process of selection approach knowledge sources for maintenance purposes.

Althoff, Hanft and Schaaf developed the idea of a Case Factory to evaluate and maintain case bases [Althoff *et al.*, 2006]. The Case Factory is based on the Experience Factory approach from software engineering. It consists of several software agents for different tasks like evaluating incoherence or modifying the case base. A central idea of the Case Factory is that an agent not only learns from his individual experience, but also learns from the experience of other agents. The idea of the Case Factory is integrated into the Knowledge Line of the SEASALT architecture. Each knowledge source, in this context CBR systems, has its own Case Factory that is responsible for maintaining the dedicated knowledge. Based on these ideas the Case Factory can be extended to maintain not the case base, but maintaining the vocabulary, the similarity measures and the adaptation knowledge, too.

The methodology to be developed within this thesis shall not only focus on CBR systems and maintenance tasks, but also considers the characteristics of distributed knowledge-based systems (e.g. the SEASALT architecture) and explanation capabilities [Roth-Berghofer *et al.*, 2005].

In addition to these methodologies, there are many different maintenance approaches for CBR systems, which should be taken into account. Ferrario and Smyth described an approach for collaborative maintenance of a case base. The feedback of several users is evaluated and an appropriate maintenance action derived [Ferrario and Smyth, 2000][Ferrario and Smyth, 2001]. Other authors like Ioannis Iglezakis [Iglezakis, 2001][Iglezakis and Roth-Berghofer, 2000], Racine and Yang [Racine and Yang, 1997][Racine and Yang, 1998][Yang and Zhu, 2001], Smyth, Keane and McKenna [Smyth, 1998][Smyth and Keane, 1995][Smyth and McKenna, 2001] or David Wilson[Wilson, 2001] described different approaches to maintain a case base. Armin Stahl describes the learning of feature weights [Stahl, 2001] and Patterson et al showed a strategy to maintain the similarity of a CBR system[Patterson *et al.*, 2000]. All these approaches are set up to maintain a single CBR system, but neither consider the use of multiple CBR systems nor the dependencies between these single systems. The maintenance approach to be developed within this thesis will consider such dependencies and will be able to combine single maintenance actions to an integrated maintenance strategy for distributed CBR systems.

3 Research Goals

Based on Section 2, four major research goals are formulated. All major research goals are split into several minor goals and tasks. The first goal is to develop a methodology, being able to define, implement and maintain distributed CBR systems based on the SEASALT architecture. This methodology contains all necessary tasks that lead to a functional multi-agent-system, as defined in the SEASALT architecture, extended with tasks covering the maintenance of CBR systems and the explanation awareness of the knowledge maintenance. The ideas underlying this methodology are described in more detail in Section 4.

The second research goal is to extend the concept of the Case Factory approach. The maintenance strategy has to be extended to cover the dependencies between several different homogenous or heterogeneous knowledge sources for cross-system maintenance and to improve the maintenance of a single CBR system. Of course for maintenance all four knowledge containers should be considered. Again a more detailed description can be found in Section 4.

The third research goal is to integrate maintenance explanation capabilities into a mulit-agent-system with distributed CBR systems. These explanations capabilities have to be considered when developing the methodology. The explanation process has to be integrated into the design and implementation of a system. For more detailed information on the explanation process see Section 4. The second and the third goal could be seen as minor goals of the first reseach goal, because of the dependencies between these goals. We decided to treat them as seperate goals, because these goals will take a large part of the research and can be divided in minor goals, too.

The fourth major research goal is to empirically evaluate the developed maintenance strategy, methodology and improved Case Factory approach. Therefore we will integrate the explanation-aware maintenance approach into the *docQuery* application and in an industrial multi-agent decisionsupport system. With the integration of the approach into *docQuery* we will show, that our methodology can be used on existing systems to extend those systems with explanation-aware maintenance capabilities. While building the new industrial multi-agent-system we will show, that our methodology can be used to build up a new multi-agent-system with explanation-aware maintenance capabilities. On both systems experts can evaluate the maintenance suggestions and related explanations.

4 Explanation-Aware Maintenance

This section describes the current status of our the ideas and goals of explanation-aware maintenance in more detail. With explanation-aware maintenance we mean an approach that enables a multi-agent-system to suggest maintenance actions to keep the knowledge in this system correct and consistent and provides explanations for the suggested maintenance actions. The suggestions can be presented to a knowledge engineer. With the related explanations the knowledge engineer should be able to understand why the multi-agent-system gives the specific suggestions. This way the knowledge engineer can make a quicker selection of the maintenance actions that should be executed and he will be able to identify potential problems in the knowledge of the multi-agent-sytem if an explanation is not comprehensible.

There are two main ideas behind this thesis proposal. The first idea is that maintenance of knowledge cannot be done per knowledge source only, but the dependencies between the knowledge sources have to be taken into account for an integrated maintenance strategy. An example for a dependency between CBR systems is the change of the vocabulary. If both systems have the same or partially the same vocabulary, a change in one system may cause a change in the other system for consistency reasons. Another example is removing one or more cases from a case base. Cases in other CBR systems could depend on one of the removed cases, so they may become inconsistent (to some degree). The system should suggest an appropriate maintenance action like removing the depending cases to keep the system's correctness/consistency.

For example, the *docQuery* application has a CBR system for regions and a CBR system for infectious diseases. Both systems have partly the same vocabulary for region names. For the region CBR system the names are part of the solution of the case structure and for the disease CBR system the names are part of the problem description of the case structure. These CBR systems have a dependency between each other. When a certain region is retrieved from the region CBR system the related infectious diseases can be retrieved from the disease CBR system using the name of the region. If the name of one region changes in the vocabulary of the region CBR system, an inconsitency between this CBR systems will occur, because the retrieved region has no match in the disease CBR system. The related infectious diseases cannot be retreived anymore. To keep the CBR systems consistent a change of the vocabulary of the disease CBR system should be suggested. Another example is the CBR system for medication. If one medicament must not be applied for a disease any longer, the system has to check if the medicament can still be applied for another disease. If the medication is not longer used the case representing the specific medicament can be deleted, ohterwise the case has to be adapted to keep the system's correctness.

The Case Factory approach and the SEASALT architecture support distributed knowledge sources in a multi-agent-system. A Case Factory can support the maintenance of the case base of a single CBR system. The original approach has to be extended to support the maintenance of the other three knowledge containers, namely vocabulary, similarity and adaptation knowledge. The original approach contains several software agents to monitor the case-base and one agent to do the necessary maintenance actions. To support all knowledge containers some more agents are needed to monitor these containers and the maintenance tasks should be split into several agents. An own maintenance agent per knowledge container is needed to support parallel maintenance of the knowledge containers. Additionally a supervising agent is required to coordinate the maintenance actions. This coordination agent is also responsible for the communication between the multiple Case Factories. Figure 1 gives an overview of an extended Case Factory.

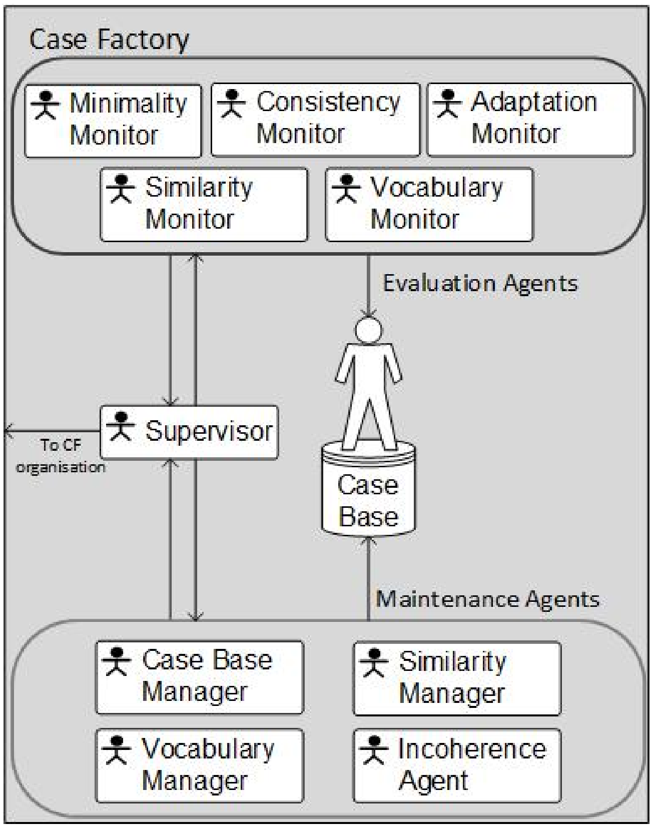

Figure 1: Extended Case Factory

Another idea is the combination of the SIAM approach and the Case Factory approach. SIAM extends the so-called *4R* cycle with two steps to a *6R* cycle. The *4R* cycle consists of four steps, called *Retrieve*, *Reuse*, *Revise* and *Retain*. In the *Retain* step a given problem is mapped to the case structure of the CBR system and the most similar case(s) is retrieved from the casebase. The *Reuse* step adapts the solution of the retrieved cases to the given problem. This adapted solutions are checked by an expert or a user of the CBR system in the *Revise* step. The result of the check can be stored in the specific case. In the last step *Retain* the retrieved, adapted and revised case can be added to the casebase. This way the CBR system can learn [Aamodt and Plaza, 1994].

The steps *Review* and *Restore* are not part of the original *4R* cycle. These steps support the monitoring and maintenance of single CBR systems [Roth-Berghofer, 2003]. Each step consists of three tasks. The steps could be mapped to a Case Factory with software agents for the defined tasks. The *Review* task contains the steps *Assess*, *Monitor*, and *Notify*. Each step could be assigned to an agent in the Case Factory. An agent responsible for the *Assess* task evaluates the knowledge of a CBR system. Another agent compares the evaluation result with the available constraints or thresholds (*Monitor*). The constraints and thresholds are defined by the knowledge engineer and stored in a so-called Maintenance Map. This Map will be described in more detail later in this section. The notifying agent decides if and whom to inform that a maintenance action is necessary. This agent sends a message to an agent at a high-level Case Factory organization. This organization is also described in more detail later. On the higher level the notifications are collected and used to create a maintenance plan.

For an example we take the CBR systems for disease and medication. An assess agent evaluates the casebase of the medication CBR systems and finds a case for a medicament that is not dedicated to a disease. These result is passed to the monitoring agent. The Maintenance Map contains the constraint that every medicament has to be dedicated to a disease. The information about the constraint violation is passed to the notify agent, which sends this information and a request for a maintenace action to a coordination agent ouside the Case Factory of the medication CBR system.

Three additional agents are responsible for the tasks of the *Restore* step. The single tasks to perform are *Suggest*, *Select* and *Modify*. The suggestions for maintenance actions are made in the high-level Case Factory organization, because the suggestions depend on the maintenance plan. The suggested actions are sent to an agent in the relevant Case Factory and this agent will select the appropriate maintenance action. It is possible, that more than one maintenance action is selected. The agent responsible for modifying the knowledge gets a message with the selected maintenance actions and executes the modification on the knowledge containers or notifys the knowledge engineer if the action needs additional input. When the modification is done or an error occurs, the agent sends a notification to the high-level Case Factory organization.

A high-level Case Factory organization is needed to control the integrated maintenance between these Case Factories. Therefore several software agents have to supervise the communication and the adherence of high-level maintenance goals. Additionally, an agent collects the suggested maintenance actions from multiple Case Factories, while another agent combines the individual maintenance actions to a maintenance plan. A Case Factory can suggest more than one maintenance action. The planning agent is also responsible for checking constraints or solving conflicts between the individual maintenance actions. A suggested action from one Case Factory can trigger a necessary maintenance action of another Case Factory, based on the dependencies between the knowledge sources. So these actions have to be integrated into the maintenance plan, too. The relevant maintenance actions are passed back by the collector agent to the relevant Case Factories.

Based on the previous example the information about the constraint violation is passed to a coordination agent in the Case Factory organization. This coordination agent checks the possibilities to repair the constraint violation. Based on the knowledge in the Maintenance Map two suggestions to repair the problems are found. The first suggestion is to delete the case to keep only cases in the casebase that could be retrieved. The second suggestion is to adapt the case and add the dedicated diseases to the problem description. Both suggestions are send back to the Case Factory of the medication CBR system. If the selection agent can decide on his own the appropriate maintenance action based on the information in the Maintenance Map, then the agent selects an action. In our example we will assume that the Maintenance Map contains the information that a case should be adapted if possible before deleting it. The selection agent will notify the knowledge engineer that additional information is needed to dedicate the medicament to a disease. The knowledge engineer can select the disease that should be dedicated to the medicament. The modify agent takes the information an adapts the problem description of the case. At last the agent sends a success or error message to the Case Factory organization. Figure 2 shows a Case Factory organization with example agents.

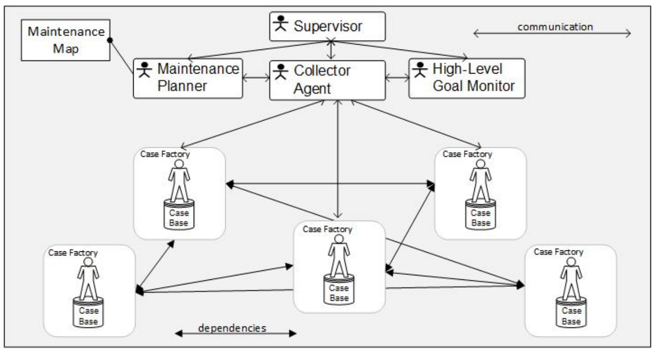

Figure 2: Case Factory organization

To coordinate the maintenance actions for the single CBR systems a Maintenance Map is introduced. This Maintenance Map is based on the Knowledge Map from Davenport and Prusak [Davenport and Prusak, 2000], that was adapted to multi-agent-systems from Bach et al. [Bach *et al.*, 2008]. In contrast to a Knowledge Map the Maintenance Map is a bidirectional graph. The vertices represent the knowledge sources in a distributed knowledge-based system and the edges represent the dependencies between the single sources. The weights of the edges could be used to describe the importance of the dependency. Additionally, the maintenance goal for a single knowledge source could be stored as well as the integrated maintenance goal for the overall system. For every maintenance goal the metrics for the empirical evaluation and the thresholds and constraints could be defined. Another idea is to store the preferred maintenance action for every knowledge source in the Maintenance Map. By this it will be possible for the relevant agent to decide quickly which maintenance action to choose. The Maintenance Map could be in XML or RDF format to share the knowledge between different systems in an easy way. The Maintenance Map should be defined and updated by the knowledge engineer.

To define and implement a distributed CBR system with explanation-aware maintenance capability a methodology is needed that contains tasks to acquire the necessary knowledge, maintenance goals, and actions to realize an integrated maintenance strategy. The SEASALT architecture implies tasks to develop a multi-agent-system with distributed knowledge-based systems (e.g. CBR systems). These tasks could be derived and organized in a methodology. For example the use of software agents implies the definition of an agent model which describes the roles and responsibilities of the agents. The resulting methodology has to be extended with tasks for maintenance and explanations.

Methodologies like DISER, DILLEBIS and INRECA contain tasks that may be integrated in the new methodology. The methodology to be developed shall be applicable to multi-agent-systems with different single CBR systems as knowledge sources. While the focus is set on using CBR systems as knowledge sources, the methodology shall contain several template tasks, that could be replaced by tasks for different knowledge-based systems. These templates have to be included in a way that substituting a template with a concrete instantiation will not affect the other tasks. Our research will focus on a methodology that can be used to define and implement a multi-agent-system with distributed CBR systems. From this methodology it could be possible to identify task that could generalized with templates to support the definition of other knowledge-based systems. An example task affecting maintenance is shown in Figure 3.

Task: define maintenance goal for Case Factory organisation

Maintenance Goal: keep overall solution correct/complete
Needed Knowledge: expert feedback, consistency rules
Metrics:-
Trigger for Evaluation: getting feedback to solution
Input: overall solution
Output: evaluation results, maintenance suggestion and explanation

Figure 3: Example task to define an overall maintenance goal

The second idea is that a CBR system should be able to explain why a maintenance action is suggested. This explanation will support the knowledge engineer's decision which maintenance action should be executed. To give a CBR system explanation capabilities a lot of knowledge is necessary. The introduced idea of an integrated maintenance strategy focuses on explanations for maintenance. The underlying research assumption here is that the minimal knowledge necessary for the explanation of the maintenance actions is the same knowledge that is necessary for the CBR system to suggest a maintenance action. It follows, that the minimal knowledge for explanations already exists in the system, if the system is able to (reasonably) suggest maintenance actions. One challenge is to identify and extract the needed part of the knowledge to formulate the explanations for a given maintenance action. For a single explanation of a maintenance action not the whole knowledge of the CBR system is needed. Another challenge is to identify and combine the knowledge of several CBR systems to explain maintenance suggestions that are necessary to keep the overall sytem correct and consistent.

Knowledge that can be used for explanations are logging information, rules, evaluation results, metrics, thresholds, etc. Additionally, knowledge can be extracted from social media like expert forums or external data sources. A scenario for extracting knowledge from sources outside the multi-agent-system can be the prohibition of a medicamant. This information can be extracted from the webside of the European Medicines Agency. Monitoring this website the multi-agent-sytem can react on new information and suggest a maintenance action for deleting or adapting a case in the medication CBR system. The same information that triggers the maintenance suggestion can be used to explain the suggestion. Such knowledge extracting has already been carried out in the scope of the SEASALT architecture [Bach, 2012] [Bach *et al.*, 2010] and can be plugged into our methodology.

5 Conclusion and Outlook

This paper describes the ideas of an integrated explanation-aware maintenance approach for distributed (knowledge-based) systems. This includes the development of an integrated maintenance strategy and the definition of a methodology with tasks that consider maintenance of distributed systems and explanations. The successful implementation of distributed knowledge-based systems in research and industrial environments to evaluate the maintenance approach and the methodology is a part of the idea, too.

The next steps are to define the goals of an integrated maintenance strategy and the extended Case Factory concept. Based on these definitions the implied tasks from the SEASALT architecture (e.g. define agent model, define knowledge acquisition, etc.) can be derived. Some of these implied tasks already exist in DISER, DILLEBIS or INRECA, other tasks like defining an agent model have to be formalized. With these task a first version of the target methodology can be set up. This methodology will be used to define and implement an explanation-aware maintenance extension for the *docQuery* application. The experience collected during this task will be used to refine and extend the methodology. The second version of the methodolgy will be used to define the mentioned industrial multi-agent-system. Both systems will be continuoisly evluated with the help of domain experts and knowledge engineers. The results will be used to improve the methodology, the maintenace strategy and the implemented multi-agent-systems.

References

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

[Althoff and Weis, 1996] Klaus-Dieter Althoff and Karl-Heinz Weis. An evaluation of the inreca cbr system. *Proceedings of the 9th German Workshop on Machine Learning*, 1:6–11, 1996.

[Althoff *et al.*, 2006] Klaus-Dieter Althoff, Alexandre Hanft, and Martin Schaaf. Case factory: Maintaining experience to learn. In *Proceedings of the 8th European conference on Advances in Case-Based Reasoning*, pages 429–442, 2006.

[Bach *et al.*, 2008] Kerstin Bach, Meike Reichle, Alexander Reichle-Schmehl, and Klaus-Dieter Althoff. Implementing a coordination agent for modularised case bases. In *Proceedings of the 13th UK Workschop on Case-Based Reasoning*, pages 1–12, 2008.

[Bach *et al.*, 2010] Kerstin Bach, Christian Sauer, and Klaus-Dieter Althoff. Deriving case base vocabulary from web community data. In *ICCBR-2010 Workshop Proceedings: Workshop on Reasoning From Experience On The Web*, pages 111–120, 2010.

[Bach, 2012] Kerstin Bach. *Knowledge Acquisition for Case-Based Reasoning Systems.* PhD thesis, University of Hildesheim, 2012.

[Bergmann *et al.*, 2003] Ralph Bergmann, Klaus-Dieter Althoff, Sean Breen, Mehmet Gker, Michel Manago, Ralf Traphner, and Stefan Wess. *Developing Industrial Case-Based Reasoning Applications: The INRECA Methodology.* Springer Verlag Berlin, 2003.

[Davenport and Prusak, 2000] Thomas H. Davenport and Laurence Prusak. *Working Knowledge: How Organizations Manage What they Know.* Havard Business School Press, 2000.

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

[Ferrario and Smyth, 2001] Maria Angela Ferrario and Barry Smyth. Distributing case-based maintenance: The collaborative maintenance approach. *Computational Intelligence*, 17(2):315–330, 2001.

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

[Iglezakis, 2001] Ioannis Iglezakis. The conflict graph for maintaining case-based reasoning systems. In *Case-Based Reasoning Research and Development: Proceedings of the Fourth International Conference on Case-Based Reasoning*, pages 263–275, 2001.

[Nick, 2005] Markus Nick. *Experience Maintenance through Closed-Loop Feedback.* PhD thesis, University of Kaiserslautern, Computer Science Department, AG Software Engineering, 2005.

[Patterson *et al.*, 2000] David Patterson, Sarabjot Anand, and John Hughes. A knowledge light approach to similarity maintenance for improving case-base competence. In *ECAI Workshop Notes*, pages 65–78, 2000.

[Racine and Yang, 1997] Kristi Racine and Qiang Yang. Maintaining unstructured case bases. In *Case-Based Reasoning Research and Development*, pages 553–564, 1997.

[Racine and Yang, 1998] Kristi Racine and Qiang Yang. Redundancy and inconsistency detection in large and semi-structured case bases. In *IEEE Transactions on Knowledge and Data Engineering*, 1998.

[Reichle *et al.*, 2011] Meike Reichle, Kerstin Bach, and Klaus-Dieter Althoff. Knowledge engineering within the application independent architecture seasalt. *Joachim Baumeister, Grzegorz J. Nalepa (Hrsg.). International Journal of Knowledge Engineering and Data Mining (IJKEDM)*, 1:202–215, 2011.

[Reichle-Schmehl, 2008] Alexander Reichle-Schmehl. Design and implementation of a software agent to coordinate the dynamic retrieval on distributed, heterogeneous case bases (in german). Bachelorarbeit, University of Hildesheim, 2008.

[Reuss, 2012] Pascal Reuss. Concept and implementation of a knowledge line - retrieval strategies for modularized, homogeneous topic agents within a multi-agent-system (in german). Master's thesis, University of Hildesheim, 2012.

[Roth-Berghofer *et al.*, 2005] Thomas Roth-Berghofer, S. Schulz, and A. Woody (Eds). Explanation-aware computing (exact 2005), technichal report fs-05-04, washington d.c., usa. Technical report, AAAI Press, 2005.

[Roth-Berghofer, 2003] Thomas Roth-Berghofer. *Knowledge maintenance of case-based reasoning systems. The SIAM methodology.* Akademische Verlagsgesellschaft Aka GmbH, 2003.

[Smyth and Keane, 1995] 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.

[Smyth and McKenna, 2001] Barry Smyth and Elizabeth McKenna. Competence models and the maintenance problem. *Computational Intelligence*, 17(2):235–249, 2001.

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

[Stahl, 2001] Armin Stahl. Learning feature weights from case order feedback. In *Case-Based Reasoning Research and Development: Proceedings of the Fourth International Conference on Case-Based Reasoning*, 2001.

[Tautz, 2000] Carsten Tautz. *Customizing Software Engineering Experience Management Systems to Organizational Needs*. PhD thesis, University of Kaiserslautern, 2000.

[Wilson, 2001] David Wilson. *Case-Based Maintenance: The Husbandry of Experience*. PhD thesis, Faculty of the University Graduate School, Department of Computer Science, University of Indiana, 2001.

[Yang and Zhu, 2001] Qiang Yang and Jun Zhu. A case-addition policy for case-base maintenance. *Computational Intelligence*, 17(2):250–262, 2001.