Maintenance of distributed case-based reasoning systems in a multi-agent system*

Pascal Reuss and Klaus-Dieter Althoff
pascal.reuss@dfki.de
klaus-dieter.althoff@dfki.de

Intelligent Information Systems Lab, University of Hildesheim
Competence Center for Case Based Reasoning, German Center for Artificial Intelligence, Kaiserslautern

Abstract. In many knowledge-based systems the used knowledge is distributed among several knowledge sources. Knowledge maintenance of such systems has several challenges to be met. This paper gives a short overview of a maintenance approach using so-called Case Factories to maintain knowledge sources and considering the dependencies between these sources. Furthermore we present a concept how our maintenance approach can be applied to a multi-agent system with several case-based reasoning systems.

1 Introduction

When maintaining the knowledge among distributed case-based reasoning (CBR) systems the dependencies between the knowledge sources are of crucial importance. For maintaining a single CBR 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. This paper describes a multi-agent system, based on the SEASALT architecture, that is extended with several agents to apply the Case Factory approach. We describe the tasks of every required agent and the communication between them. In addition we present the required agents for the explanation capabilities. Section 2 describes related work to knowledge maintenance. In Section 3 the agents required for applying the Case Factory approach to a multi-agent system are described. In Section 4 a short conclusion is given.

1.1 SEASALT architecture

The SEASALT (Shared Experience using an Agent-based System Architecture Layout) architecture is a domain-independent architecture for extracting, analyzing, sharing, and providing experiences [[5]]. The architecture is based on the Collaborative Multi-Expert-System approach [1][2] and combines several software engineering and

 In: T. Seidl, M. Hassani, C. Beecks (Eds.): Proceedings of the LWA 2014 Workshops: KDML, IR, FGWM, Aachen, Germany, 8-10 September 2014, published at http://ceur-ws.org

artificial intelligence technologies to identify relevant information, process the experience and provide them via an interface. The knowledge modularization allows the compilation of comprehensive solutions and offers the ability of reusing partial case information in form of snippets. Figure 1 gives an overview over the SEASALT architecture.

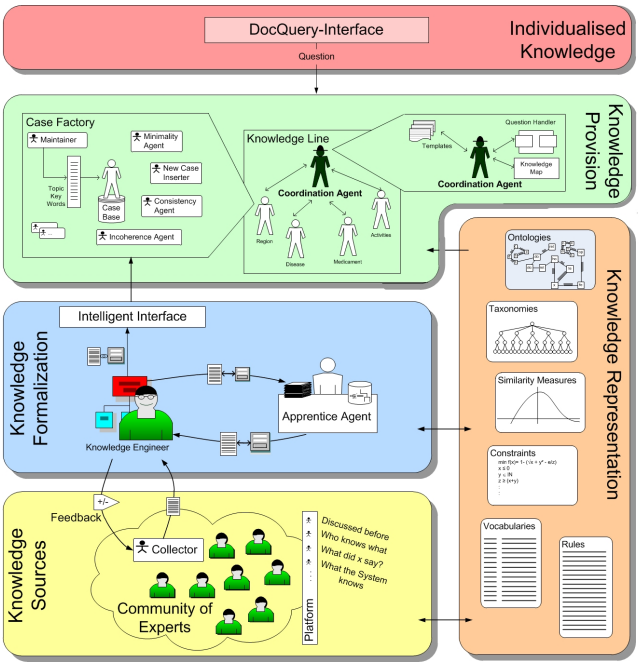

Fig. 1. Overview of the SEASALT architecture

The SEASALT architecture consists of five components: the knowledge sources, the knowledge formalization, the knowledge provision, the knowledge representation, and the individualized knowledge. The knowledge sources component is responsible for extracting knowledge from external knowledge sources like databases or web pages and especially Web 2.0 platforms. These knowledge sources are analyzed by so-called Collector Agents, which are assigned to specific Topic Agents. The Collector Agents collect all contributions that are relevant for the respective Topic Agent's topic [5]. The knowledge formalization component is responsible for formalizing the extracted knowledge from the Collector Agents into a modular, structural representation. This formalization is done by a knowledge engineer with the help of a so-called Apprentice Agent. This agent is trained by the knowledge engineer and can reduce the workload for the knowledge engineer [5]. The knowledge provision component contains the so called Knowledge Line. The basic idea is a modularization of knowledge analogous to the modularization of software in product lines. The modularization is done among the

individual topics that are represented within the knowledge domain. In this component a Coordination Agent is responsible for dividing a given query into several sub queries and pass them to the according Topic Agent. The agent combines the individual solutions to an overall solution, which is presented to the user. The Topic Agents can be any kind of information system or service. If a Topic Agent has a CBR system as knowledge source, the SEASALT architecture provides a Case Factory for the individual case maintenance. [5][4] The knowledge representation component contains the underlying knowledge models of the different agents and knowledge sources. The synchronization and matching of the individualized knowledge models improves the knowledge maintenance and the interoperability between the components. The individualized knowledge component contains the web-based user interfaces to enter a query and present the solution to the user.[5]

2 Related work

This section contains related work from other authors with focus on the maintenance of the knowledge containers of CBR systems and the maintenance of distributed knowledge in CBR systems. There exist several approaches to maintain the knowledge containers of a CBR system. For the maintenance of a case base various strategies were developed for example by [9], [10], [12], [13], [18], [17], [19] and [22]. [20] and [11] describe approaches to maintain the similarity measures within a CBR system. All this approaches are set up to maintain knowledge containers of a single CBR system. They neither consider the use of multiple CBR systems nor the dependencies between the knowledge containers of different CBR systems. All mentioned maintenance strategies could be applied within a Case Factory, but have to be embedded in an overall maintenance strategy managed by the Case Factory Organization.

Geissbuhler and Miller describe in their paper an approach for maintaining distributed knowledge bases in a clinical decision support system called WizOrder. Contrary to our approach, the maintenance in the WizOrder system is not done by one knowledge engineer, but by many different users of the system, like house staff, physicians, and nurses. The knowledge sources in the decision support system are heterogeneous and not homogenous as intended in our approach. Therefore many different tools for maintenance are used, each one with a specific interface for the respective user. The local knowledge bases are maintained by the users and an expert integrates the maintenance actions into the central knowledge base called knowledge library. From this knowledge library the cumulative changes are provided to the local knowledge bases. While this is done by human users and experts within the WizOrder system, in our approach we use software agents to suggest maintenance actions and central planning and supervising agents to generate a maintenance plan. This plan has still to be checked by a human knowledge engineer. [8]
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. When we compare our approach to theirs, our agents could be seen as users, that gives feedback and suggest maintenance actions. A Case Factory, maintaining one

CBR system could be compared to the collaborative maintenance. One difference between the approaches is that our approach is extended with maintenance capabilities for several CBR systems.[6][7]

3 Maintenance of distributed case-based reasoning systems

This section gives a short overview over the idea of the Case Factory (CF) and the Case Factory Organization (CF). Then the software agents for the realization of the CF and CFO are described. At last the software agents for the explanation capabilities are described.

3.1 Agents of the Case Factory

Three types of software maintenance can be distinguished: corrective, adaptive and perfective maintenance. Corrective maintenance deals with processing failures, performance failures and implementation failures. Processing failures are situations like abnormal termination of an application. Performance failures deals with situations where the application violates defined performance constraints like to long response time. Implementation failure can lead to processing and performance failures, but may also be have no effect on the system. Adaptive maintenance deals with changes in the environment of an application and aims at avoiding failures caused by the change of an application environment. Perfective maintenance cover all actions that are performed to eliminate processing inefficiencies, enhance performance or improve the maintainability. This type of maintenance aims at keeping an application running at less expense or running to better serve the users needs [21]. [16] defines the knowledge maintenance of CBR systems as the combination of technical and associated administrative actions that are required to preserve the knowledge of a CBR system, or to restore the knowledge of the system to provide the intended functionality. This maintenance actions include also actions to adapt an CBR system to environment changes and enhance the performance.

The SEASALT architecture supports the maintenance of a CBR system with the help of a Case Factory. The original idea is from Althoff, Hanft and Schaaf [3] and the concept was extended by Reuss and Althoff [14]. The CF approach and the SEASALT architecture support the maintenance of distributed knowledge sources in multi-agent systems and the CF is intended to perform corrective maintenance as well as adaptive and perfective maintenance. Feedback from users about false solutions may lead to corrective maintenance actions, while the evaluation of the knowledge in a CBR system may lead to adaptive or perfective maintenance. The extended CF supports the maintenance of a single CBR system. It contains several software agents responsible for the evaluation and the maintenance of the case-based reasoning knowledge containers. This knowledge containers were introduced by [15].

The idea behind the Case Factory approach is maintenance of knowledge sources should consider the dependencies between the knowledge containers in a CBR system and the dependencies between the knowledge containers of different CBR systems.

There are dependencies between the vocabulary and the case base in a single CBR system or between case bases in different CBR systems. Changing the knowledge in one knowledge container may cause inconsistencies. Therefore additional maintenance actions may be necessary to restore the consistency of the knowledge.

To apply the CF approach to the a multi-agent system with CBR system nine agents are required: four monitoring and evaluation agents, each one responsible for one knowledge container (case base, vocabulary, similarity, and adaptation), and four maintenance agents, each one responsible for processing individual maintenance actions for the respective knowledge container. We propose an individual monitoring and evaluation agent for each knowledge container to process the monitoring and evaluation tasks in parallel. In addition it will be possible to activate and deactivate the monitoring and evaluation of a knowledge container during runtime by starting or shutting down the associated agent without affecting the monitoring and evaluation of the other knowledge containers. The last new agent is a supervising agent that coordinates the monitoring and evaluation of the knowledge containers and the processing of maintenance actions. In addition the agent communicates with the high-level Case Factory Organization. Figure 2 shows these agents in a multi-agent system.

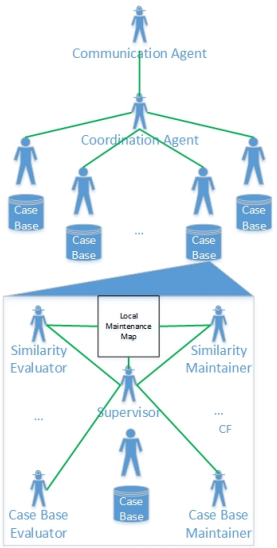

Fig. 2. Multi-agent system with Case Factory agents

In the following the tasks, permissions and responsibilities of the agent roles are described in GAIA notation [23]. Protocols and activities define the communication with other roles and the tasks a role can perform. The permissions are used to describe the knowledge a role has access to and the knowledge a role can change or generate. At last the responsibilities are used to describe the life cycle of a role. It is defined in which

order the protocols and activities are performed and if there are repetitions of protocols or activities. A (*) means, that a protocol or activity is performed 0 to n times, a (+) that a protocol or activity is performed 1 to n times. The exponent at the end of the liveness responsibilities describes the times the the whole process is performed. ω means it is repeated endlessly.

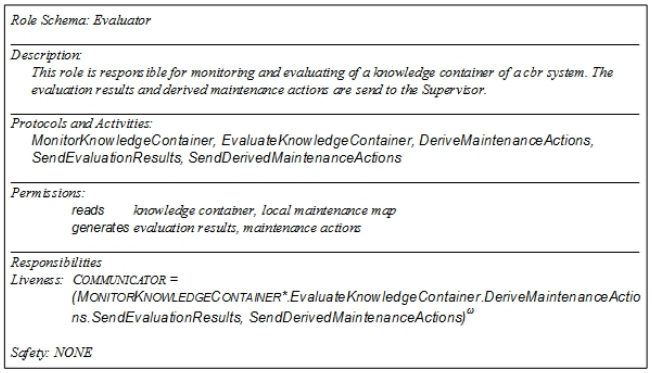

Fig. 3. Role schema Evaluator in Gaia notation

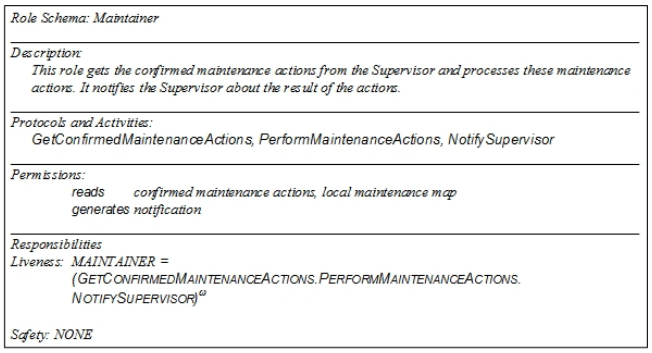

Fig. 4. Role schema Maintainer in GAIA notation

Both generic roles are specialized for the specific agents and generic terms are substituted with the concrete knowledge container. Both roles have access to a local maintenance map, which contains information about available and preferred evaluation strategies and maintenance actions as well as evaluation metrics to compare the results to the maintenance goals. This way several evaluation strategies can be defined and applied to an agent.

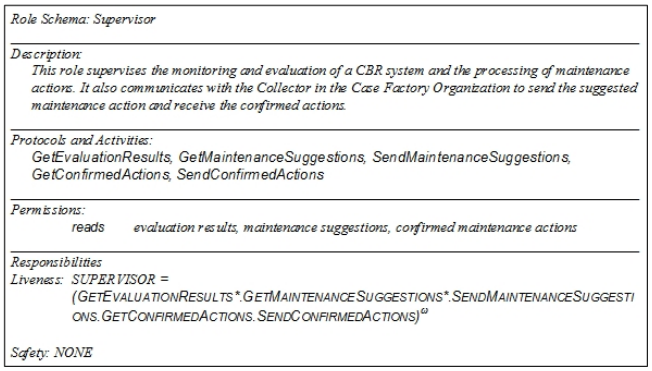

Fig. 5. Role schema Supervisor in GAIA notation

3.2 Agents of the Case Factory Organization

While a Case Factory is able to maintain a single CBR system a high-level Case Factory Organization is required to coordinate the actions of all Case Factories and take the dependencies between the single CBR systems into account. This CFO consists of several additional software agents to supervise the communication between the Case Factories and the adherence of high level maintenance goals. Additionally, agents collect the maintenance suggestions from the Case Factories and derive a maintenance plan from all single maintenance suggestions. The agents are also responsible for checking constraints or solving conflicts between individual maintenance suggestions. In addition, a maintenance suggestion may trigger follow-up maintenance actions based on the dependencies between the CBR systems. The concept of the CFO allows to realize as many CFs and layers of CFOs as required. A multi-agent system can be divided into layers and each layer can have its own Case Factory Organization. This way a hierarchy of CFOs can be established that is scalable and supports multi-agent systems with many agents and layers. [14]

Each required Case Factory Organization consists of four software agents. A Collector Agent, a Maintenance Planning Agent, a Goal Monitoring Agent and a Team Supervisor Agent. For the assumed MAS only one Case Factory Organization level is required. Figure 6 shows the multi-agent system with the the additional agents for the Case Factory Organization.

Inside the CF agents evaluating the knowledge containers and derive maintenance suggestions from the result with the help of the local maintenance map (1). The results and the derived maintenance actions are send to the supervisor (2). The supervisor passed the maintenance actions to the collector (3). This collector gets the derived maintenance actions from all Case Factories and sends them to the goal monitoring agent. The goal monitoring agent is responsible for checking the maintenance actions against constraints from the team maintenance map. If no constraints are violated the maintenance actions are sent to the maintenance planner (5). This agent generates a plan from the maintenance actions. During the planning process it is possible to generate new maintenance actions based on the dependencies between different CBR systems. The

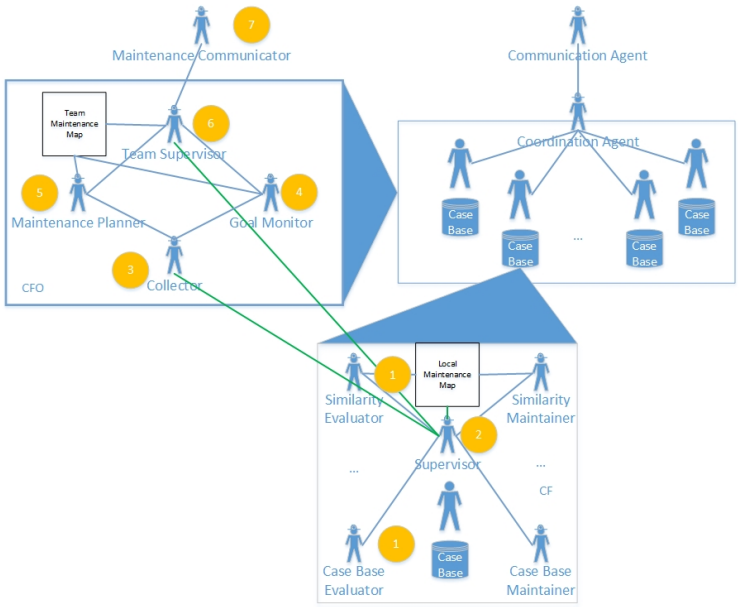

Fig. 6. docQuery multi-agent system with Case Factory Organization agents

maintenance plan is sent to the team supervisor (6). This agent checks the plan against constraint violation like the goal monitor does for individual actions. The checked plan is sent to the maintenance communicator and shown to a knowledge engineer (7). The knowledge engineer checks the plan and confirms the maintenance actions to be performed. He can also eliminate actions from the plan. The confirmed plan is sent back to the team supervisor in the CFO, the supervisor in the CF and the single maintenance actions to the maintaining agents.

Our concept for the Case Factory Organization includes explanation capabilities of the maintenance actions and the maintenance plan. The idea is to provide a set of explanations to support the knowledge engineer's understanding of the suggested maintenance plan and single actions. The idea is to use explanation templates that are filled with logging information. These templates consists of several text modules in human natural language. This way we try to use the systems logging information to generate human readable explanations.

To achieve this goal, the multi-agent system has to log all communication and actions of all agents, as well as evaluation results, feedback, constraint checks, and denied maintenance actions. From this logged information explanations should be extracted and combined for each maintenance action and the maintenance plan itself. Three additional roles are required to provide simple explanations: Logger, Logging Supervisor, Explainer. For each role at least one agent in the docQuery multi-agent system will be implemented. For several roles like the Logger or the Evaluator more than one instance is required. Some of the described roles and the respective agents can be combined in agent teams. For example, for a Case Factory a team of four Evaluators, four Main-

tainers and one Supervisor is requiresd. Adding a new Case Factory will require the creation of nine software agents. Other roles like the Logger or the Explainer and its respective agents can be added as single agents. This way the multi-agent system has a high scalability and agents can be created and removed based on the tasks the single agents or the agent team are designed for.

Figure 7 shows the multi-agent system with all agents for the CF, CFO, and explanations. In the figure only the communication with the new agents is illustrated.

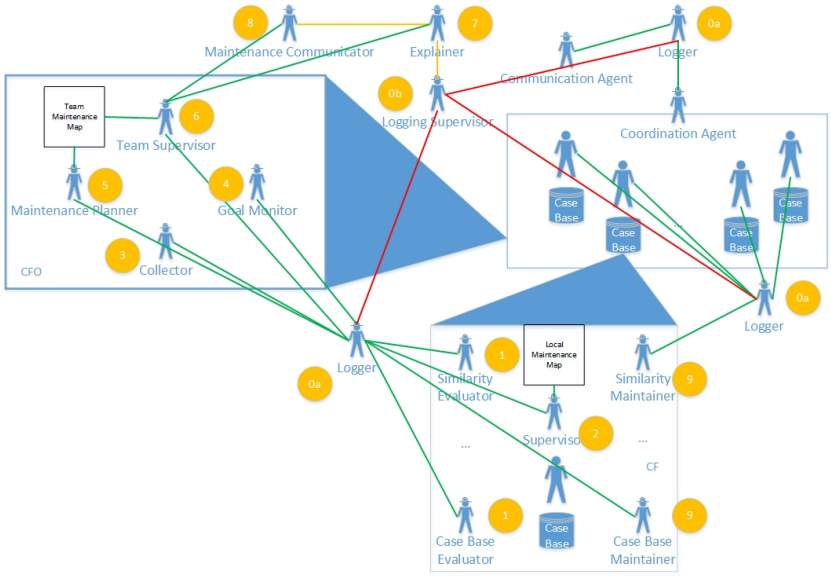

Fig. 7. MAS with CF, CFO and explanation agents

Several agents are responsible for logging the communication and performed tasks of the agent in the multi-agent system (0a) and the logged information are send to the logging supervisor (0b). These information are used to generate explanations for suggested maintenance actions. Steps 1 till 6 are the same as described above. In addition, the checked plan is sent to the explanation agent (7). This agent uses the logged information to enrich the maintenance plan with explanations. The enriched plan is sent to the maintenance communicator and shown to a knowledge engineer (8). The knowledge engineer checks the plan and confirms the maintenance actions to be performed. He can also eliminate actions from the plan. The confirmed plan is sent back to the team supervisor in the CFO, the supervisor in the CF and the single maintenance actions to the maintaining agents (9).

4 Summary and Outlook

In this paper we presented the concept for a multi-agent system in the travel medicine domain with software agents for distributed maintenance with explanation capabilities. We gave an short overview of the Case Factory and Case Factory Organization and described the required tasks of the individual agent roles. The roles do not describe any concrete implementation of tasks or communications. The realization of the described concept and the implementation of the agents within a multi-agent system is the next step in our research. We will implement the single agents and agent teams as well as evaluation and maintenance strategies and evaluate the extended multi-agent system.

References

1. Althoff, K.D.: Collaborative multi-expert-systems. In: Proceedings of the 16th UK Workshop on Case-Based Reasoning (UKCBR-2012), located at SGAI International Conference on Artificial Intelligence, December 13, Cambride, United Kingdom. pp. 1–1 (2012)
2. Althoff, K.D., Bach, K., Deutsch, J.O., Hanft, A., Mänz, J., Müller, T., Newo, R., Reichle, M., Schaaf, M., Weis, K.H.: Collaborative multi-expert-systems – realizing knowledge-product-lines with case factories and distributed learning systems. In: Baumeister, J., Seipel, D. (eds.) KESE @ KI 2007. Osnabrück (Sep 2007)
3. Althoff, K.D., Hanft, A., Schaaf, M.: Case factory: Maintaining experience to learn. In: Proceedings of the 8th European conference on Advances in Case-Based Reasoning. pp. 429–442 (2006)
4. Althoff, K.D., Reichle, M., Bach, K., Hanft, A., Newo, R.: Agent based maintenance for modularised case bases in collaborative mulit-expert systems. In: Proceedings of the AI2007, 12th UK Workshop on Case-Based Reasoning (2007)
5. Bach, K.: Knowledge Acquisition for Case-Based Reasoning Systems. Ph.D. thesis, University of Hildesheim (2013), dr. Hut Verlag Mnchen
6. Ferrario, M.A., Smyth, B.: A user-driven distributed maintenance strategy for large-scale case-based reasoning systems. In: ECAI Workshop Notes. pp. 55–63 (2000)
7. Ferrario, M.A., Smyth, B.: Distributing case-based maintenance: The collaborative maintenance approach. Computational Intelligence 17(2), 315–330 (2001)
8. Geissenbuhler, A., Miller, R.A.: Distributing knowledge maintenance for clinical decision-support systems: The "knowledge library" approach. In: Proceedings of the AMIA Symposium. pp. 770–774 (1999)
9. Iglezakis, I.: 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 (2001)
10. Iglezakis, I., Roth-Berghofer, T.: A survey regarding the central role of the case base for maintenance in case-based reasoning. In: ECAI Workshop Notes. pp. 22–28 (2000)
11. Patterson, D., Anand, S., Hughes, J.: A knowledge light approach to similarity maintenance for improving case-base competence. In: ECAI Workshop Notes. pp. 65–78 (2000)
12. Racine, K., Yang, Q.: Maintaining unstructured case bases. In: Case-Based Reasoning and Development. pp. 553–564 (1997)
13. Racine, K., Yang, Q.: Redundancy and inconsistency detection in large and semi-structured case bases. IEEE Transactions on Knowledge and Data Engineering (1998)
14. Reuss, P., Althoff, K.D.: Explanation-aware maintenance of distributed case-based reasoning systems. In: LWA 2013. Learning, Knowledge, Adaptation. Workshop Proceedings. pp. 231–325 (2013)

15. Richter, M.M.: Introduction. chapter 1 in case-based reasoning technology - from foundations to applications. lnai 1400, springer (1998)
16. Roth-Berghofer, T.: Knowledge maintenance of case-based reasoning systems. The SIAM methodology. Akademische Verlagsgesellschaft Aka GmbH (2003)
17. Smyth, B.: Case-based maintenance. In: Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (1998)
18. Smyth, B., Keane, M.: 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. pp. 377–382 (1995)
19. Smyth, B., McKenna, E.: Competence models and the maintenance problem. Computational Intelligence (2001)
20. Stahl, A.: 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)
21. Swanson, E.B.: The dimensions of maintenance. In: Proceedings of the 2nd International Conference on Software Engineering. pp. 492–497 (1976)
22. Wilson, D.: Case-Based Maintenance: The Husbandry of Experience. Ph.D. thesis, Faculty of the University Graduate School, Department of Computer Science, University of Indiana (2001)
23. Wooldridge, M., Jennings, N., Kinney, D.: The gaia methodology for agent-oriented analysis and design. Autonomous Agents and Multi-Agent Systems 3, 285–312 (2000)