Agent Based Maintenance for Modularised Case Bases in Collaborative Multi-Expert Systems

Klaus-Dieter Althoff, Meike Reichle, Kerstin Bach, Alexandre Hanft, Régis Newo

Intelligent Information Systems Lab
University of Hildesheim
Marienburger Platz 22, 31141 Hildesheim, Germany
althoff|reichle|bach|hanft|newo@iis.uni-hildesheim.de

Abstract. This paper features SEASALT, an intelligent information system architecture that follows the example of collaborating human experts. Modularised knowledge is provided in a number of case bases that offer their topic expertise which is used to combine knowledge for individual queries. We describe how collaborative multi-expert systems can be instantiated using CBR technology to cover the experts' roles and agent technology in order to enable their collaboration. We focus on the maintenance of cases in such multi-case-base systems regarding inter-case base dependencies. We evaluate our approach based on the real-life application of travel medicine and show how the requirements of an information system on travel medicine can be fulfilled using our hybrid, agent based CBR system architecture.

Keywords: CoMES; Collaborative Multi-Expert Systems; SEASALT; hybrid, agent based CBR system; case base maintenance; distributed case bases; inter-case base dependency; travel medicine

1 Introduction

Nowadays, in our mostly goal-oriented need for information, we rely on modern information systems, which we expect to be good enough to provide us (quickly) with all the information we need. One of the main requirements of those systems is that they should be sufficiently smart to support their users requiring as little effort as possible. Hence they should offer knowledge-intensive services that are able to adapt and expand their knowledge. In this paper we present an approach called CoMES (Collaborative Multi-Expert Systems) for developing intelligent information systems for delivering such knowledge-intensive services and supporting knowledge workers. Here we especially focus on experience (case-specific knowledge, cases) which is known to be the success-critical knowledge in these kinds of applications.

As its name already suggests, CoMES is related to the expert system[1] approach [1]. While prominent problems of the latter were that they mostly ignored the task of

[1] In this paper we do not differentiate between the notions of *expert system* and *knowledge-based system*.

embedding expert systems in real business processes and lacked a good solution for overcoming the so-called knowledge acquisition bottleneck, CoMES tries to overcome these problems by integrating various approaches from artificial intelligence and software engineering. For example, the software product-line approach [2] is applied to the knowledge within an expert system with the goal of decomposing the knowledge into different modules (which will be called knowledge-line). We use the multi-agent system approach [3] to model and realise this. As case-specific knowledge is a prominent part of the knowledge to be captured and reused, these software agents are mostly implemented as case-based reasoning (CBR) systems. By this we also want to enable these agents to learn from their own and others' experience. Here we also benefit from the already achieved integration of CBR in Experience Factory, an approach for (organizationally) learning from experience by means of human-enacted (business) processes [4]. While in this integration CBR helped to model these processes more precisely, the Experience Factory provides business processes that are suited for evaluating and/or maintaining case bases. We now try to enact these processes using software agents; an approach which we call Case Factory [5]. It introduces software agents working on the continuous improvement of the case base. As a guideline for this, processes enacted by human agents become candidates for being partially automated using software agents. A first step is having a software agent assist a human agent with the software agent learning from the experience of this human coach. Following steps include realizing an – at least – two-layered architecture that enables "complicated" learning strategies by means "simple" case-based learning. In this paper we focus on how to maintain distributed case bases using the case factory and knowledge-line approaches. Another important aspect of CoMES is the integration of various collaborating experts, as known from social software approaches of web 2.0. That is, we try to learn from the expertise of other peers and community experts.

As a first evaluation of our approach we present the docQuery real-life application [6], which we currently develop together with mediScon worldwide and the German Aerospace Center (DLR). It covers the travel medicine domain and is currently being realised using the SEASALT (Sharing Experience using an Agent-based System Architecture LayouT) architecture, our first instantiation of CoMES.

In chapter 2 we will present the docQuery application in more detail. Following a review of related work chapter 3, we will focus on the maintenance of multiple case bases for realising knowledge-lines in chapter 4. We then evaluate our approach using the docQuery application on travel medical data in chapter 5. Finally some conclusions and an outlook on potential future work are presented chapter 6.

2 Application Domain: docQuery[2]

During the last decade travelling to different places, experiencing new cultures and meeting new people all over the world has become more and more popular. In

[2] docQuery is a project in co-operation with mediScon worldwide (www.mediscon.com) and TEMOS (http://temos-network.com/), the telemedicine project of the Institute of Aerospace Medicine at the German Aerospace Centre – DLR (http://www.dlr.de/me/)

preparation for a healthy journey it is important to get high quality and reliable information on travel medicine prevention. Travel medicine is the specialised area of medicine that deals with medical issues like diseases, vaccinations, etc., which might occur before, during and after a journey. In fact, it focuses on what happens to people when they change their regular environment, for example when travelling by car, train or airplane to different places [6].

There are already many websites and web forums in which travel medicine information can be found (e.g., which vaccinations should be administered when someone plans to travel to a given country[3]). The main drawback of such websites and web forums is that they usually do not contain all necessary medical information and the traveller has to visit many pages to receive all the information he needs. Thus, it is a difficult and time-consuming task to gather all the information for a travel destination. Furthermore, the editors of the sites are mostly unknown and travellers cannot evaluate whether the given information is trustworthy, complete and/or correct.

We aim to remedy these problems and will create docQuery, an intelligent information system on travel medicine in co-operation with a team of certified doctors of medicine with a strong background in travel related medicine. This offers us the possibility to establish a community for experts in which they can exchange their knowledge on their expertise (e.g., coping with chronic illnesses during a journey) and get new information from their colleagues.

Based on our SEASALT architecture, we will implement the docQuery application [6] which will provide the travellers with travel medicine information tailored to suit their journey. Queries to the system contain data like travel period, destination, age of the traveller. We will be able, by the means of docQuery, to create a community to exchange knowledge and offer a multi-expert system on travel medicine tasks. Integrating a community challenges our system to organise the knowledge using both, software and human agents.

3 Related Work: Case Base Maintenance

Maintenance in CBR systems is a continuous process during the life-cycle of such systems [7], especially if the system has to manage dynamically changing content.

At first we refer to known problems and used measures to guide maintenance. CBR maintenance mainly takes place on the case base as stated by Iglezakis and Roth-Berghofer [8]. This means maintaining a case base by adding, deleting and editing cases according to certain aims, for instance avoiding identical cases or reducing the size of case bases. Because many CBR systems insert a new case during each retain step of the well-known 4-RE-CBR Cycle [9], many CBR maintenance approaches are supervising the performance of case base retrieval [10] to deal with the utility problem [11]. Others focus on competence models [11−13] to guide the addition and deletion of cases. In this context Smyth and McKenna define the maintenance problem [12] as how to optimise the performance of a CBR system. Smyth and Keane

[3] http://wwwn.cdc.gov/travel/default.aspx

propose a competence preserving case deletion policy [11] depending on the properties coverage and reachability. Since we do not retain cases after each query, we have to pay attention to those properties mostly while inserting cases.

Racine and Yang [14] suggest different measures for the evaluation of huge case bases but concentrate on consistency and define intra-case case consistencies as well as inter-case consistencies, which are defined as case-base properties across two or more cases. Reinartz et al. [15] pick up on the ideas of Racine and Yang and define the case and case base properties *correctness, consistency*, *uniqueness* and *minimality*. Using these properties Iglezakis builds up a *conflict graph* [16] and uses a cost function on that graph to decide which case should be modified to *restore* the case base quality.

Apart from mostly technical approaches the maintenance should also be embedded in a more organised process. To organise maintenance Roth-Berghofer and Iglezakis [17] enhance the 4-RE-CBR Cycle with two new steps *review* and *restore* towards a six-step model (called six-RE-cycle) which handles maintenance explicitly. They notice that maintenance should be done beyond the usual working task on a CBR application. Based on this work Roth-Berghofer [7] introduces a more general methodology called SIAM which guides *Setup*, *Initialization*, *Application* and *Maintenance*.

Maintenance should not only be seen from the more analytical perspective like Roth-Berghofer does, but also as an interaction process with humans within organizations. Althoff, Tautz and others integrate the Experience Factory, an organizational approach for continuously learning from experience, and CBR [18-20]. They define Experience-based Information Systems (EbIS) as a type of information system that contains experiences and is able to handle a (more or less) continuous „stream of experience“.

The integration of the quality improvement paradigm (QIP), the task-method-decomposition of CBR, and the components-of-expertise approach (CoE), were used as a base to develop the DISER method for systematically constructing CBR systems [21]. Nick introduces the DILLEBIS method [22] which extends and revises DISER with respect to experience lifecycle models and the reuse of respective EbIS patterns. Patterns he introduced include, for instance, a basic feedback loop for situated experience or a recursive feedback loop. DISER/DILLEBIS is in contrast to other methods for CBR system development/maintenance like INRECA [23] or SIAM [7] not restricted to CBR systems, but abstracts from underlying technologies and is appropriate for other AI technologies.

Nick and Althoff [24] point out that in order to implement maintenance successfully, the EbIS has to be designed from the start as a maintainable system, because for long-lived systems in general maintainability is a crucial requirement and quality criterion. They demonstrate that maintenance needs methodological support for planning, managing and conducting maintenance which also leads us to the idea of maintenance through a case factory. All these considerations are also part of the CoMES approach.

4 Using Multiple Case Bases to Realise Knowledge-Lines

Knowledge-Lines were first introduced in [5] and further detailed in the so called CoMES (Collaborative Multi-Expert-Systems) approach, described in [1]. The underlying idea is to use the principle of product lines as it is known from software engineering [2] and apply it to the knowledge in a knowledge-based system (knowledge-lines), thus splitting rather complex knowledge in smaller, reusable units. The provision and reuse of these units is realised by different intelligent topic agents, which are realised as individual CBR systems. Each of these agents' case bases is maintained by a case factory, an organizational unit that emulates the experience factory approach [4] from software engineering.

The CoMES approaches first practical instantiation is the SEASALT architecture [25] that picks up on knowledge-lines and case factories. This architecture is also used to build the docQuery system. Within the SEASALT architecture the information system's knowledge domain – in the case of docQuery travel medicine – is modularised into different topics or sub-domains. Each topic is represented by a topic agent, a CBR system equipped with information on one particular aspect of travel medicine (for instance geographic information or travel related diseases) and a case factory taking care of its case base.

The information stored in the respective case bases is acquired from an online discussion forum for a community of domain experts that is equipped with collector agents. These agents collect experts' contributions that are relevant for the collector's respective topic agent and pass them to a knowledge engineer for formalisation, so they can be handled in a structured CBR system. The formalisation is done by one or more human domain experts in the beginning but shall be at least partially automated as more training data become available. While this human knowledge engineer will increase the system's overall expenses (be it financial expenses or volunteer work), we consider the direct connection to the domain experts community to be valuable enough with regard to the system's accuracy and timeliness as to warrant these expenses. A query to the overlying information system will be handled by a coordination agent that subsequently queries the different topic agents according to a request strategy, which indicates which topic agents' answers give constraints for another topic agent's query, and a knowledge map that indicates where/how the individual topic agents can be contacted. The coordination agent will then combine the information it received from the individual topic agents into an answer, using ready-prepared templates, that is finally passed back to the user. This approach is comparable to the negotiated case retrieval approach presented in [26], but relies on a more strict partition with regard to the case bases' content, which allows a fixed definition of combination rules (the request strategy, for example "Find out the region first, use the region to find relevant diseases, ask for chronic illnesses, use diseases and preconditions to retrieve medicaments") and does not require negotiation between the individual topic agents.

The approach of using several case bases within a case-based reasoning (multi-case-base reasoning, MCBR) system exists within the CBR community for several years already [27]. In the beginning case bases were partitioned mostly for performance reasons, in order to allow a faster case retrieval within systems with a very large case base. In other applications cases can not be centralised within one single

case base for privacy or maintenance reasons, for instance in a situation where different parties each run their own individual case base and do not want to share their entire content or maintenance with other involved parties. Also, Leake und Sooriamurthi [28] present experimental evidence that, given data from different sources, treating these as individual case bases, being coordinated using a dispatcher, which selects appropriate knowledge sources/case bases and also combines returned cases, yields to better results than simply merging all available data into one centralised case base.

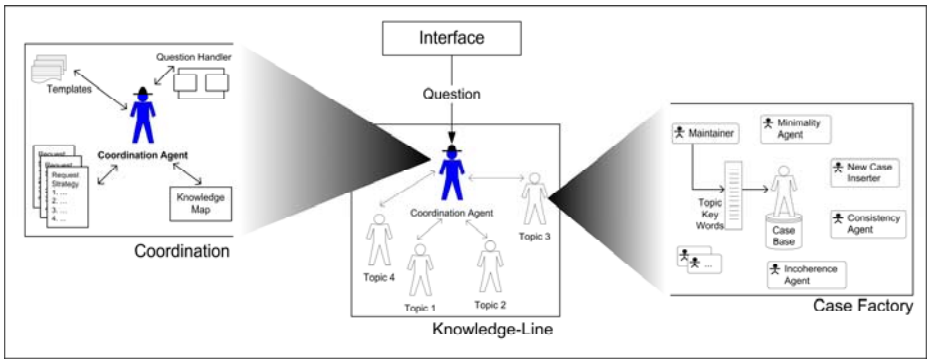

Fig. 1. The Knowledge-Line. Left: The coordination agent and its different tools. Centre: The knowledge-line including the coordination agent and different topic agents. Right: The Case Factory of a topic agent with the different maintenance agents.

The modularisation in sub-domains instead of simply partitioning a case base into several smaller ones with the same domain/case format has several different advantages. Firstly the individual case bases are easier to maintain with regard to the correctness of their contents, since they represent a more simple knowledge domain. Also breaking up the rather complex domain of travel medical advisories into more simple sub-domains that are than recombined as needed, gives the whole information system more flexibility. Provided the appropriate combination rules are available, the contents of the individual case bases can also be combined into cases that have not yet been presented to the system, as long as they adhere to the respective combination rules. Further, not all knowledge domains that are included in an information system on travel medicine (geographic information, diseases, medicines and their active pharmaceutical ingredients, travel activities) require the same type of maintenance and are subject to the same amount of change over time. While it is for instance no problem to keep a rather simple domain such as countries and regions as minimal and consistent as possible, the domain of travel related diseases is better served by including as many cases as possible, even if some of them are very similar. By splitting the knowledge domain into these smaller sub-domains, they can all be maintained in a way that is best suited for the respective sub-domain.
In our approach the employment of multiple case bases is motivated by the individual case bases' different maintenance needs. We intend to implement the case bases' maintenance following the case factory approach, which means that it will be carried out by several agents, each performing a single and rather simple task. These different

tasks will each ensure one individual case base property, as described for instance in [15]. Analogous to the case base properties presented there, we will employ individual agents to ensure the case bases' consistency, uniqueness/minimality, incoherence (avoiding cases that are too similar) while other agents will be used for adding new cases, following an intelligent case-addition policy such as [13] or creating generalisations of similar cases. Distributing these different maintenance tasks among individual agents allows for a better fine tuning of the individual tasks and also for better adjusting the balance between conflicting case base maintenance aims such as minimality (and thus better performance) and competence [29].

Since our modularisation happens with respect to different sub-domains, the individual case bases that are the result of this modularisation are not absolutely independent of each other. Instead they have a net of dependencies between them that indicates what other case bases are affected by changes in one individual case base. These dependencies also affect some of the aforementioned maintenance tasks and split them into two different kinds: those which have to take other case bases into consideration and those which do not. A case factory agent that performs the task of maintaining a case base's uniqueness, minimality, incoherence or consistency can do this without knowing about other case bases. An agent that inserts new cases or adds new data to existing cases is a different matter. For instance if a new case is inserted into the *diseases* case base, or an existing disease breaks out in a new region, the inserting agent has to check, whether every risk region indicated in the respective disease's *regions* attribute is actually included in the *country* case base's underlying ontology[4], otherwise this *diseases* case will never be retrieved. Another example of such dependencies would be, if one of the regions can not be associated, a domain expert will have to be contacted and asked for specifications, since we assume that in this case there is either a simple typing error, a synonymous name for a region, or the new region will have to be added to the ontology by a domain expert. The same is true for new data being added to the *medicament* case base. Here the *area of application* has to yield at least one result in the *disease* case base, otherwise the new data have again to be passed to a domain expert for a review. Although this approach is rather maintenance-intensive, our medical application scenario requires this very conservative case addition strategy in order to preserve the system's overall accuracy and the case factory approach allows us to realise this strategy and also adapt to new dependencies, should they arise.

5 Maintenance on Travel Medical Data

Travel medical data contain information about countries, diseases, medications, vaccinations as well as descriptions, guidelines, and experiences. Therefore the knowledge in docQuery will be provided in case bases and each case base will contain one specific topic with its own domain model and maintenance agents/jobs. However, following the modularised structure of knowledge in docQuery, CBR will be used for each

[4] The *country* case base will use a geographical ontology for the description of geographic terms, which can be integrated in our architecture, because we will use e:IAS of empolis [30] which offers the usage of ontologies during the retrieval process.

individual topic agent providing information. Aiming at higher accuracy each case base will serve its own topic and the case format will exactly fit for the type of knowledge. Furthermore the case bases will contain similarity measures, adaptation knowledge and the vocabulary to represent and retrieve cases. In travel medicine it can be dangerous to use CBR for the whole set of information, because the combination of medications, vaccinations, side effects, contraindications, etc. regarding the traveller's health history have to be correct, without any contradicting information. Instead of that we will apply CBR for each topic and do the combination of the responses afterwards using the constraints given in the response sets. Each issue handled in a case base will be provided using CBR methods and the strength of CBR, finding similar information on a given topic, will ensure a higher quality of information provision.

In this section we will exemplify four docQuery case bases that are each representing one topic and explain the dependencies between them. The selected case bases are examples to explain our approach and for the implementation of docQuery there will be at least six more case bases.

The case base *country* will contain specific country information a traveller has to consider planning a journey. Furthermore the information will be separated in the sections a traveller has to pay attention to before, during, and after the journey. The country information also includes the required vaccinations and additional information, for example guidelines for a healthy journey.

The case base *disease* holds more than 100 diseases considered in a travel medical consultation. It concentrates on diseases that might affect a traveller on a journey, for instance Malaria, Avian Influenza, or Dengue. A disease in this case base is characterised by general information on the disease, how to avoid the disease, how to behave if one has had the disease before, and how to protect oneself.

The third case base we will introduce is *medicament* with details about medicaments and their area of application (diseases, vaccinations, age, etc.). Basically it contains information about active pharmaceutical ingredients, effectiveness, therapeutic field, contraindication, interdependences, and the countries in which those medicaments are approved. In *diseases* we do not store information on chronic illnesses, because they will be modelled in their own case base. Instead we focus on diseases which can affect travellers during their journey.

The fourth case base will hold *activities* which are used within docQuery to provide safety advice for intended activities when planning a journey. For travellers, activities are the major part of their journey, but may involve certain risks for which safety advice is needed and furthermore while asking for their plans they usually describe their activities which we can use to provide better guidance. Examples of such activities are diving, hill-climbing or even swimming.

Complete travel medical information will contain knowledge of all four case bases enhanced with descriptions, guidelines, and previous experiences. The combination of the information retrieved from each case base will be done by a Coordination Agent as it can be seen in the SEASALT architecture (Fig. 1). The coordination agent will request each agent and based on the agents' response and the given information by the traveller the next request containing all constraints to another topic agent will be created and send.

Inter-case base dependencies arise from features that influence each other even if they are contained in different case bases. Fig. 2 shows the case formats for four case bases and their inter-case base dependencies on which we will give some examples.

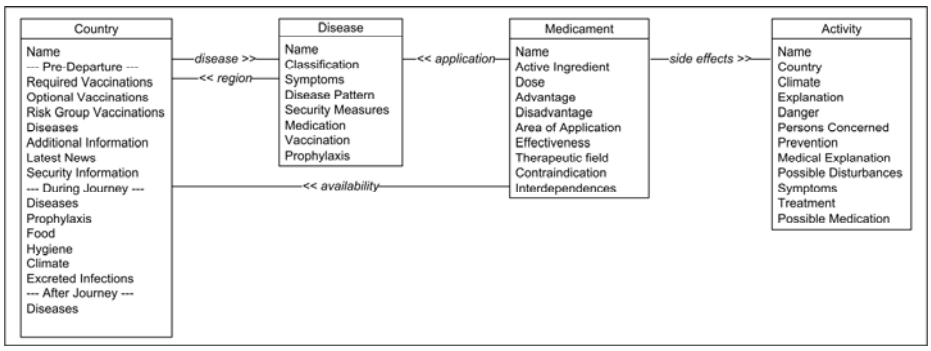

Fig. 2. Case formats for four case bases that provide knowledge for the topic agents: country, disease, activity and medicament and their inter-case base dependencies. This figure shows what kind of information is stored in the case bases and gives some examples of how dependencies can be defined.

An example for a travel medical request might be that a German family will travel in March to Alor to dive and afterwards they will travel around Bali by car. Using the given information all four case bases have to be requested to find appropriate information. The first information given by the traveller is the fact that they will go to Alor and Bali in March. Since Alor and Bali are not countries, first we have to figure out where those places are situated: both are Indonesian islands. Furthermore, the travelling period will be March, which will be in the rainy season and now, the inter-case base dependencies *disease* and *region* have to be applied, because travelling to Indonesia, especially in the region of Alor/Flores/Komodo/Sumbawa/Sumba will require a malaria prophylaxis. For that reason information on malaria has to be provided and therefore the home destination is required, because depending where travellers come from, medicaments can be recommended, or not. Also the dependency *availability* describes if a certain medicament is approved in that particular country. Medicaments can cause *side effects* which might influence the planned activities during a journey. An example for such side effects is that Doxycyclin Monohydrat[5] as malaria prophylaxis causes light toxicity in some cases. Light toxicity can occur using some kinds of medicaments and afterwards people's skin reacts hypersensitive to sun. As a consequence, the traveller should try to avoid the sun and use high sun blocker.

In this case the traveller has to be advised against using the medicament and all side effects have to be mentioned and an explanation of how to cope with the side effects must be provided. Of course the choice has to fit the disease, but it must also

[5] Doxycyclin Monohydrat is one medicament for malaria prophylaxis, but it is not approved for malaria prophylaxis in every country. Nevertheless the World Health Organization (WHO) and other countries, like the USA or Australia, recommend it for malaria prophylaxis and especially in Lombok (Indonesia) it is recommended and applied.

be considered whether the traveller can take it or not (*application*). If the travellers in our example take this medicament as malaria prophylaxis they have to be careful to get information how they should behave and which side effects might occur while diving, swimming, etc. We mentioned above one side effect and a possible safety measure, but usually there are many more and docQuery is generally aiming to give travellers the opportunity to follow their activities instead of forbidding them. Furthermore we are trying to give the travellers the opportunity to pursue their activities safely by following medical advice.

As the given example shows our application domain is very complex and we decided to follow the modularised knowledge structure to achieve high quality case bases (for example country or disease) and in a second step combine the results of each field of expertise. The dependencies between the case bases will help us to create valid combinations and maintain all relevant data. In our architecture the dependencies are modelled in the request strategies and support the coordination agent to send requests and constraints to each topic agent. When new cases have to be inserted, updated, or deleted the inter-case base dependencies have to be regarded.

We expect *country* information to change very frequently, because these cases do not only contain geographic but also security information. The content of the case bases *disease* and *medicament* will change less often and the most static cases will be in *activity*, because its characteristics do not vary once they are covered. Also the information in our case bases can be inserted individually, for instance new medicaments can be inserted in their case base as they are discussed in the expert's web community without having to change the other case bases.

6 Conclusion and Outlook

In this paper we gave an introduction to the CoMES approach and its resulting architecture for intelligent information systems, SEASALT. Within SEASALT we focused on knowledge modularisation, which splits a complex domain into several sub-domains, each represented by an individual case base, and how such a modularised knowledge base can be maintained using intelligent agents. We first gave an overview on already existing work in CBR maintenance and detailed our own position in respect to the individual approaches.

Following this, we introduced the SEASALT architecture and its main components, the *Knowledge-Line* and the *Case Factories*. Subsequently we discussed how a system of interdependent case bases can be maintained, adopting existing approaches, transferring them to a multi-agent setup, and adapting them to the additional requirements of inter-case base dependencies. Next we evaluated our methodology by applying it to the real-life application scenario of travel medicine. We will research, whether the SEASALT architecture will help to develop the travel medicine application within a shorter amount of time and/or with less effort and/or with a more complete knowledge base etc., than other approaches like, for instance, a "standard CBR" approach.

Once the docQuery system has been fully implemented we will use the SEASALT architecture to realise other, related application scenarios such as FLOSSWALD, an

information system based on free/open source software and SIMOCOSTS, a model for simulating cognitive processes in order to analyse how human beings act in critical situations by emphasising which coping strategies they use [25].

References

1. 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 Knowlegde-Product-Lines with Case Factories and Distributed Learning Systems, In: Baumeister, J., Seipel, D. (eds) Proceedings of the 3rd Workshop on Knowledge Engineering and Software Engineering (KESE 2007), September 2007. University of Osnabrück, (2007)
2. Van der Linden, F., Schmid, K., Rommes, E. (eds.) Software Product Lines in Action – The Best Industrial Practice in Product Line Engineering. Springer Verlag, Berlin (2007)
3. Weiss, G.: Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence. The MIT Press (1999).
4. Basili, V.R., Caldiera, G., Rombach, H. D.: Experience Factory. In: Marciniak, J.J (ed.) Encyclopedia of SE, Vol 1, pp. 469–476. John Wiley & Sons, New York (1994)
5. Althoff, K.-D., Hanft, A., Schaaf, M. Case Factory – Maintaining Experience to Learn. In: Göker, M., Roth-Berghofer, T. (eds.) In: Advances in Case-Based Reasoning – Proceedings of the 8th European Conference, ECCBR 2006, Fethiye, Turkey, September 2006. LNAI, vol. 4106, pp. 429–442. Springer Verlag, Berlin Heidelberg (2006)
6. Bach, K.: docQuery – A Medical Information System for Travellers. Internal project report, September 2007, University of Hildesheim (2007)
7. Roth-Berghofer, T.: Knowledge Maintenance of Case-Based Reasoning Systems – The SIAM Methodology, dissertation at University of Kaiserslautern, available as DISKI 262, Akademische Verlagsgesellschaft GmbH, Berlin (2003)
8. Iglezakis, I., Roth-Berghofer, T.: A survey regarding the central role of the case base for maintenance in case-based reasoning. In: Minor, M. (ed.) ECAI Workshop Notes, Humboldt-Universität zu Berlin, pp. 22–28 (2000)
9. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications 7(1), 1994, pp. 39–59 (1994)
10. Leake, D. B. Wilson, D. C.: Remembering Why to Remember: Performance-Guided Case-Base Maintenance. In: Blanzieri, E. Portinale, L.: Proceedings of the 5th European Workshop, EWCBR 2000 Trento, Italy, September 2000. LNAI, vol. 1998, pp. 83–99. Springer Verlag, Berlin Heidelberg (2000)
11. Smyth, B., Keane, M. T.: Remembering to Forget: A Competence Preserving Deletion Policy for Case-Based Reasoning Systems. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 1995), Montréal, Québec, Canada, pp. 377–382. Morgan-Kaufmann, San Francisco (1995)
12. Smyth, B. McKenna, E.: Competence Models and the Maintenance Problem. In: Goebel, R., Greiner, R., Lin, D. (eds.) Computational Intelligence – Special Issue on Maintaining CBR Systems, Volume 17(2), pp. 235–249. Blackwell Publishers, Cambridge, Massachusetts (2001)
13. Zhu, J. Yang, Q.: Remembering to add: Competence–preserving case addition policies for case base maintenance. In: Proceedings of the International Joint Conference in Artificial Intelligence (IJCAI 99), Stockholm, pp. 234–241. Morgan Kaufmann, San Francisco (1999)
14. Racine, K. Yang, Q.: On the consistency management of large case bases: the case for validation. In: Proceedings of the AAAI–96 Workshop on Knowledge Base Validation, American Association for Artificial Intelligence (AAAI) (1996)

15. Reinartz, T., Iglezakis, .I, Roth-Berghofer, T.: On quality measures for case base maintenance. In: Proceedings of the 5th European Workshop on Case-Based Reasoning (EWCBR). LNAI, vol 1898, pp. 247–259. Springer-Verlag, Berlin Heidelberg (2000)
16. Iglezakis, I.: The conflict graph for maintaining case-based reasoning systems. In: Proceedings of the 4th International Conference on Case-Based Reasoning (ICCBR). LNAI, vol. 2080, pp. 263–275. Springer-Verlag, Berlin Heidelberg (2001)
17. Roth-Berghofer, T., Iglezakis, I.: Six steps in case-based reasoning: Towards a maintenance methodology for case-based reasoning systems. In: Professionelles Wissensmanagement: Erfahrungen und Visionen (includes the Proceedings of the 9th German Workshop on Case-Based Reasoning (GWCBR)), pp. 198–208. Shaker-Verlag, Aachen (2001)
18. Althoff, K.-D., Wilke, W.: Potential Uses of Case-Based Reasoning in Experience Based Construction of Software Systems and Business Process Support. In: Bergmann, R., Wilke, W. (eds.) Proc. of the 5th German Workshop on Case-Based Reasoning (GWCBR'97), LSA-97-01E, Centre for Learning Systems and Applications, University of Kaiserslautern, pp. 31–38 (1997)
19. Tautz, C., Althoff, K.-D.: Using Case-Based Reasoning for Reusing Software Knowledge. In: Leake, D. B., Plaza, E.: Case-Based Reasoning Research and Development – 2nd International Conference (ICCBR'97), Providence, RI. LNCS vol. 1266, pp. 156–165. Springer Verlag, Berlin Heidelberg (1997)
20. Althoff, K.-D., Mänz, J. & Nick, M.: Integrating Case-Based Reasoning and Experience Factory: Case Studies and Implications. In: Proc. 28th German Conference on Artificial Intelligence Workshop on Knowledge Engineering and Software Engineering, pp. 1–12. Technical Report 8/2005, University of Koblenz-Landau, ISSN 1860-4471 (2005)
21. Tautz, C.: Customizing Software Engineering Experience Management Systems to Organizational Needs. PhD thesis, University of Kaiserslautern, Germany. Published by Fraunhofer IRB Verlag, Stuttgart (2001)
22. Nick, M.: Experience Maintenance Through Closed-Loop Feedback. PhD Thesis, University of Kaiserslautern, Germany. Published by Fraunhofer IRB Verlag, Stuttgart (2005)
23. Bergmann, R., Althoff, K.-D., Breen, S., Göker, M., Manago, M., Traphöner, R., Wess, S.: Developing Industrial Case-Based Reasoning Applications. LNAI, vol. 1612, 2nd ed. Springer-Verlag, Berlin Heidelberg (2003)
24. Nick, M., Althoff, K.-D.: Designing Maintainable Experience-based Information Systems. In: Womser-Hacker, C. and Mandl, T.: Proceedings of Hildesheimer Evaluierungs- und Retrieval Workshop (HIER 2005), September 2005. UVK-Verlag, Konstanz (2005)
25. Bach, K., Reichle, M., Althoff, K.-D.: A Domain Independent System Architecture for Sharing Experience, In: Proceedings of LWA 2007, Workshop Wissens- und Erfahrungsmanagement, pp. 296–303, Martin-Luther-University Halle-Wittenberg, ISBN 978-3-86010-907-6 (2007)
26. Prasad, M. V. N., Lesser, V. R., Lander, S. E.: On retrieval and reasoning in distributed case bases. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics 1995 (1995)
27. Plaza, E., McGinty, L.: Distributed case-based reasoning, The Knowledge Engineering Review, Vol. 1–4. 2005, Cambridge University Press, Cambridge (2005)
28. Leake, D. B., Sooriamurthi, R.: Dispatching Cases versus Merging Case-Bases: When MCBR Matters. In: Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference, FLAIRS-2003, pp. 129–133 (2003)
29. Leake, D. B., Wilson, D. C.: Guiding Case-Base Maintenance: Competence and Performance. In: Proceedings of the 14th European Conference on Artificial Intelligence (ECAI), Workshop on Flexible Strategies for Maintaining Knowledge Containers, pp. 47–54. (2000)
30. empolis GmbH. Technical White Paper e:Information Access Suite. Technical report, empolis GmbH, September 2005.