Implementing a Coordination Agent for Modularised Case Bases

Kerstin Bach, Meike Reichle, Alexander Reichle-Schmehl,
and Klaus-Dieter Althoff

Intelligent Information Systems Lab
University of Hildesheim
Marienburger Platz 22, 31141 Hildesheim, Germany
[lastname]@iis.uni-hildesheim.de

Abstract. The work presented in this paper focuses on the composition of retrieval results returned by distributed case bases. We describe how different knowledge sources can be accessed using an abstract description language and in which way we handle the resulting heterogeneous information. The realisation of this query agent is a part of SEASALT, an architecture for intelligent information systems, which follows the example of collaborating human experts and further on provides an architecture that contains all aspects of knowledge utilisation. Within SEASALT knowledge is provided in distributed knowledge sources represented by a number of case based agents. The coordination agent presented in this paper uses those knowledge sources to combine information to compose information into individual answers. We evaluate our approach based on the real-life application of travel medicine and show how the retrieval in distributed case bases can be coordinated and executed.

Keywords: Collaborative Multi-Expert Systems; Coordination Agent, Knowledge Provision, Distributed Case Bases

1 Introduction

In this paper we present an implementation of flexible knowledge provision based on distributed, heterogeneous knowledge sources that can be accessed in different ways. We combine retrieval results of several Case-Based Reasoning (CBR) systems embedded in a multi-agent system as a part of the realisation of Collaborative Multi-Expert-Systems (CoMES) presented in [1]. The novelty of our approach is the use of heterogeneous case bases for representing a modularised complex knowledge domain. There have been other approaches using partitioned and/or distributed case bases, but still differ from our approach. A description of these approaches is also included in this paper.

The work we present focuses on one aspect of the SEASALT architecture [2], the knowledge provision, which points out how to access, combine and provide knowledge of different sources. SEASALT (Sharing Experience using an Agent-

based System Architecture LayouT) proposes an architecture for intelligent information systems with several cross-linked case bases that are used to store information on different aspects of a complex knowledge domain and are filled with information mined from the online communication of a community of experts.

Our approach does not only feature CBR systems, we are also able to deal with data bases, web services, or other knowledge sources that can be accessed in the WWW. Nevertheless, in this paper we focus on modularised case bases as they were presented in [3]. Using the CoMES approach to implement complex applications we benefit from the easier maintainability of modularised case bases and their mapping to certain areas of expertise. Complex application domains follow different aspects and like in large companies we also have experts (in our approach Topic Agents) that are working together in order to handle complex problems or questions. An example for complex application domains is travel medicine from which the examples in this paper are derived.

This paper is structured as follows: In section 2 we describe our application domain followed by the explanation of the knowledge provision within SEASALT and the core idea of distributed case bases or knowledge sources in section 3. Section 4 presents the implementation of the Coordination Agent based on the requirements given by the SEASALT architecture as well as on usability aspects, followed by a detailed description of a Knowledge Map that holds metadata on the knowledge sources and the communication interface that enables the realisation within an agent framework. Related work to our approach is pointed out in section 5 and an evaluation of the current status of the implementation as well as future work in this area is presented in section 6. The final section summarises the work presented in this paper and outlines our next steps in this area.

2 The docQuery Application Domain

Travel medicine is an interdisciplinary speciality concerned with the prevention, management and research of health problems associated with travel, and covers all medical aspects a traveller has to take care of before, during and after a journey. For that reason it covers many medical areas and combines them with further information about the destination, the activities planned and additional conditions which also have to be considered when giving medical advice to a traveller. Travel medicine starts when a person moves from one place to another by any mode of transportation and stops after returning home without diseases or infections. A typical travel medical application could be a German family who wants to spend their Easter holidays diving in Alor to dive and afterwards they will travel around Bali by car. In case a traveller gets sick after a journey a travel medicine consultation might also be required. First of all we will focus on prevention work, followed by information provision during a journey and information for diseased returnees. Since there are currently no sources on medical information on the World Wide Web that are authorized by physicians and/or experts, we aim at filling this gap by providing trustworthy travel medical information for everybody.

The research project within which this work has been done is supported by mediScon worldwide, a Germany based company with a team of physicians specialized on travel medicine and TEMOS[1], a telemedical project of the Institute of Aerospace Medicine at the German Aerospace Center (DLR). Together we are developing docQuery, an intelligent information system on travel medicine that provides relevant information for each traveller about their individual journey.

We are realising docQuery based on the SEASALT architecture and our modularised case bases are implemented using the empolis Information Access Suite (e:IAS) [4], which is an industrial strength tool based on CBR. Currently, we have identified seven different case bases that we use to retrieve information: they contain information about countries, diseases, medications, vaccinations as well as descriptions, guidelines, and experiences. The modularised knowledge in docQuery is provided using CBR and each case base contains one specific topic with its own domain model, rules, similarity measures and cases. Each case base will serve its own topic and the case format will exactly fit the type of knowledge, which enables a higher accuracy of the whole collaborative system.

The combination of medicaments used for vaccinations and the treatment of chronic diseases can cause side effects or contraindications; thus it is necessary to obtain the correct health history of a traveller and to recommend a solution without any contradicting medicaments, information or advises. Therefore we do the combination of the responses afterwards using the constraints given in the response sets.

3 Knowledge Provision in SEASALT

In SEASALT the knowledge provision task is carried out by a so called Knowledge Line that contains a Coordination Agent and a number of Topic Agents that each covers one homogeneous area of expertise. The idea of the Knowledge Line concept originates in software product lines as they are described in [5], which focus on modularisation of tasks in order to create adaptable and flexible software (products). In terms of SEASALT we use the modularisation aspect to combine knowledge based on numerous different and homogeneous knowledge sources implemented as CBR software agents. Each CBR agent is covering a certain topic (in our example travel medicine that would be region, disease, medicament, activities) and is implemented as a CBR-System maintained by a Case Factory [6]. The Case Factory approach is based on the Experience Factory that uses CBR in order to coordinate software engineering projects [7].

Fig. 1 depicts a Knowledge Line and its components that are providing and maintaining knowledge in a Case Factory (left) and combining knowledge using additional information (right) in order to answer questions or provide information. We assume that an architecture based on Topic Agents is much easier to maintain than having one monolithic case base, especially when dealing with rather complex domains. Each Topic Agent is equipped with a Case Factory that contains its case base on which

1 TElemedicine for a MObile Society, see http://www.temos-network.org

retrieval queries are executed as well as agents that generate new cases, keep the case base consistent, remove incoherent cases, create and maintain knowledge models, etc. The Coordination Agent is the centre of the Knowledge Line and orchestrates the Topic Agents to enable the combination of the retrieval results.

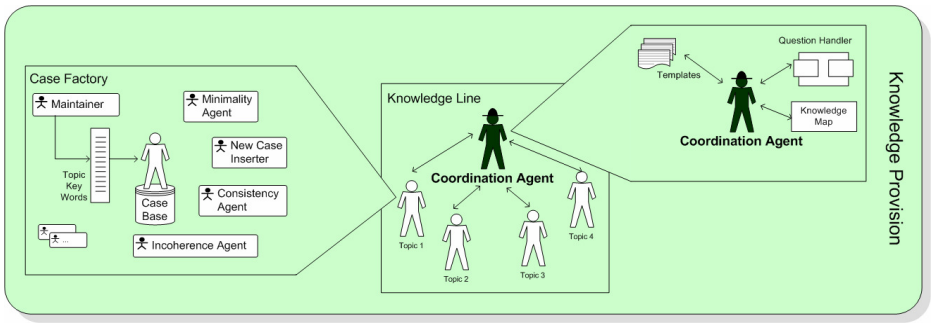

Fig. 1. Knowledge Provision within SEASALT: Knowledge Line consisting of one Coordination Agent and several Topic Agents each based on a Case Factory of its own

Even if we have different kinds of Topic Agents and their according Case Factories, the behaviour of some Case Factory agents (like the new case inserter) can be reused in other Case Factories of the same Knowledge Line. We differ between agents that handle general aspects and are contained in any Case Factory and agents that are topic-specific and have to be implemented individually. General Case Factory agents usually focus on the performance or regular tasks like insertion, deletion, merging of cases. Topic specific Case Factory agents are for example agents that transfer knowledge between the knowledge containers [8] or define certain constraints and usually they have to be implemented for an individual topic considering its specifications or fulfilling domain dependent tasks.

The Knowledge Line retrieves its information, which is formalised by a Knowledge Engineer and/or machine learning algorithms, from knowledge sources like databases, web services, RSS-feeds, or other kinds of community services and provides the information as a web service, in an information portal, or as a part of a business work flow. The flexible structure of the knowledge line allows designing applications incrementally by starting out with one or two Topic Agents and enlarging the knowledge line, for example with more detailed or additional topics, as soon as they are available or accessible.

4 Implementation of the Coordination Agent

The implementation of the Coordination Agent followed a set of requirements that were derived from the SEASALT architecture description itself and from the implementation and testing of the Topic Agents.

4.1 Requirements

During the design phase of the Coordination Agent the following requirements were identified:

- The case representations of the Topic Agents differ from each other as well as the agents' respective location might vary. This requires flexible access methods that are able to deal with distributed locations, different kinds of result sets and possibly also different access protocols.
- Some Topic Agents require another Topic Agent's output as their input and thus need to be queried successively, others can be queried at any time. In order for the Coordination Agent to be able to obey these dependencies they need to be indicated in the Knowledge Map in an easily comprehensible way.
- Based on the dependencies denoted in the Knowledge Map the agent needs to be able to develop a request strategy on demand. This request strategy should also be optimisable with regard to different criteria such as the Topic Agents' response speed, the quality of their information, the possible economic cost of a request to a commercial information source and also possible access limits.
- In order to guarantee the quality of the final result of the incremental retrieval process there needs to be a possibility to control what portion of the result set is passed on to the subsequent Topic Agent. This portion should be describable based on different criteria such as the number of cases or their similarity.
- In order to allow for higher flexibility and a seamless inclusion in the SEASALT architecture the functionalities need to be implemented in an agent framework.

4.2 Knowledge Map

Firstly, in order for the Coordination Agent to be able to navigate the different knowledge sources a format for the Knowledge Map had to be designed and implemented. Since the dependencies between Topic Agents can take any form, we decided to implement the Knowledge Map as a graph where each Topic Agent is represented by a node and directed edges denote the dependencies. The case attributes that serve as the next Topic Agent's input are associated with the respective edges. The optimisation criteria are indicated by a number between 0 (worst) and 100 (best) and are represented as node weights. In order to be able to limit the portion of the result that is passed on to the next node we implemented four possible thresholds, namely

- the total number of cases to be passed on
- the relative percentage of cases to be passed on
- the minimum similarity of cases to be passed on
- the “placement” with regard to similarity of the cases to be passed on. (For instance the best and second best cases.)

An example graph from the docQuery application can be seen in Fig. 2.

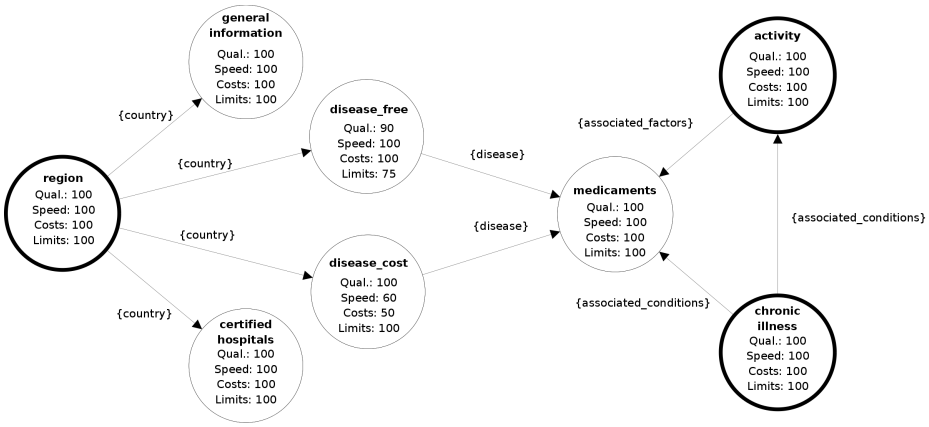

Fig. 2. An example graph based on the docQuery application

According to our example introduced in the beginning of this paper the region agent would return a case including the information that Alor and Bali are Indonesian islands. Based on this information (i.e. Country = Indonesia) queries for general safety information about this country, diseases that can be contracted in the country, and certified (international standard) hospitals at the destination are initiated. In this example there are two agents offering that information, a free one[2] with information of lesser quality and a commercial one[3] with information of higher quality. The retrieved diseases (Malaria, Yellow Fever, Diphtheria, Tetanus, Hepatitis A, Typhoid Fever, etc.) are then subsequently used to query the medicaments agent for recommendable vaccinations and medicaments that can be taken at the location. This query returns an initial list of recommendable medicament candidates. Further on, the information given by the user (Activities = "diving" and "road trip") is used to request information from the activity agent defining constraints for medicament recommendations (e.g. Activity = "Diving" => Associated_factors = "high sun exposure") which are then again used to query the medicaments agent. In this example a query for Counter_Indication = "high sun exposure" would return, among others, the Malaria prophylaxis Doxycyclin Monohydrat, which would then be removed from the initial list of recommended medicaments. Also, if specified, the influences of chronic illnesses on recommended medicaments and planned activities are queried. The combined information from all Topic Agents is compiled into an information leaflet using ready prepared templates. (*"When travelling to Indonesia, please consider the following general information: ... Certified hospitals can be found in the following places: ... A journey to Indonesia carries the following risks: ... We recommend the following medicaments: ... These medicaments are not recommended because of the following reasons ..."*)

The Knowledge Map itself is stored as an XML document. We use RDF as the wrapper format and describe the individual nodes with a namespace of our own. More details concerning the XML-Format can be found in [9]. Based on the knowledge

2 The cost 100 denotes a minimal price, that is 0,-

3 The price is medium high, thus the cost value is 50, an agent with a higher cost would have an even lower cost value.

map we then use a modified Dijkstra algorithm [10] to determine an optimal route over the graph. The algorithm is modified in such a way that it optimises its route by trying to maximise the arithmetic mean of all queried nodes. In the case of a tie between two possible routes the one with the lesser variance is chosen.

4.3 Communication Interfaces

In order to address the requirement of flexible access to the heterogeneous Topic Agents the communication interface was implemented as an abstraction layer to access various kinds of Topic Agents. Although we mostly use CBR Systems as Topic Agents we also want to be able to seamlessly incorporate external knowledge sources. Because of this the interface is implemented in specialised classes (one per access method) which can be individually instantiated at runtime when constructing the internal representation of the knowledge map.

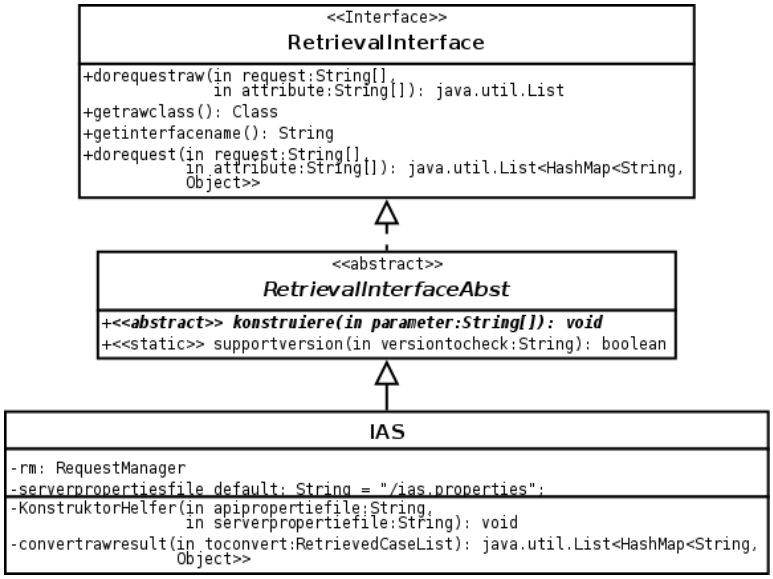

Fig. 3. A UML diagram of the connection interface

Different kinds of Topic Agents might also need a different number of parameters to initiate a connection, an external Topic Agent that is based on a data base might for example need host and port, username and password, while others, such as the e:IAS based agents of the docQuery application just need the path to a configuration file containing more detailed information on how to access the Topic Agent. Because of this we needed to implement a way to pass different numbers of parameters to the constructor of the class. Therefore the specialised query classes do not implement the designed interface directly, but extend an abstract class in which a constructor takes an array of strings as parameter. Furthermore, the abstract class declares an abstract method, which is called by the constructor and is implemented by the specialised classes. In this method specialised classes implement the connection instantiation.

Fig. 3 shows a UML class diagram with the designed interface, the abstract class and one class implementing the access method to one kind of topic agent (in this case the access to e:IAS.)

The Interface was designed to be as easily feasible as possible and therefore offers only the bare minimum of functionalities needed to query Topic Agents. The method `dorequest()` is used to query a Topic Agent, the result is a regular `java list` containing `java maps` including the attribute names of the respective query results and their specific values.

Another requirement towards the interface was high usability. It should be easy for developers to use it to access different Topic Agents and it should be easily expandable by new specialised classes to query not yet supported kinds of Topic Agents. Because of this we decided to use mainly parameter types and return types that java developers should be familiar with. The results of queries are represented by simple lists of maps (in contrast for example to representing them in an XML document which can be queried using XPath).

As the containing agent framework we chose Jade [11] since it is also Java based, so the implemented functionalities could easily be integrated.

5 Related Work

Currently and in the past there have been several approaches that feature(d) a dispatching or coordination agent between different knowledge sources. Basically our approach varies because we are combining different kinds of information. For example Ontañón and Plaza [12] presented an approach in which solutions for the same problem have to be coordinated and selected. In comparison to our approach only one of the retrieved cases has to be selected instead of using the retrieved cases to compile one holistic solution. Further on, in Leake's and Sooriamurthi's approach [13], due to complexity of the application domain and operability, a dispatching agent for selecting the best result out of a number of retrieval results returned by several CBR systems has been introduced. But all case bases contained the same type of cases (same case representation), so the dispatching agent's task was different from ours. It had to select the best case instead of using the retrieved solution as a part of the overall solution. Combining parts of cases in order to adapt given solutions to a new problem has been introduced by Redmond in [14] in which he describes how snippets of different cases can be retrieved and merged into other cases, but in comparison to our approach, Redmond uses similar case representations from which he extracts parts of cases in order to combine them. His approach and the knowledge provision in SEASALT have in common that both deal with information snippets and put them together in order to have a valid solution. Generally speaking a lot of the approaches we use on knowledge sources could also be applied to a set of cases from one case base. Analogous to our knowledge sources being of different quality and trustworthiness one could also have one case base with different cases being of different quality and trustworthiness. So our notion of knowledge source attributes is comparable in that regard and thus benefits from advances in this field of CBR (like the recent work of Briggs and Smyth [15]). However, from our point of view, the

graph-like representation of the knowledge sources and its use in the composition of the final results cannot be easily applied to a set of cases from one case base. The reason for this is, that in our approach one knowledge source covers one domain and we can thus make assumptions on its semantics (the region case base will always return a country, a country is always associated with illnesses, etc.). Having a set of cases from one case base using one case representation, this assumption does not hold and thus we cannot define the dependencies required in our graph.

The implementation of the Coordination Agent within the Knowledge Line can be compared to a service oriented architecture approach, but it is realised within an agent framework, because software agents within a framework can fulfil more flexible and autonomous tasks [16] than web services. Another approach that adjoins to ours is the concept of negotiating agents [17]. For example in our application there are several competing aims. The traveller wants do to as many of the planned activities as possible but chronic illnesses or medicaments' side effects might prevent him or her, furthermore the traveller wants to vaccinate against as many diseases as possible but some vaccinations are incompatible. Instead of solving or at least optimising these conflicts centrally using a coordination agent another possible solution might be to have the respective agents negotiate the optimal solution among themselves without a mediator. However, in our travel medical approach, the dependencies are straight forward and do not require any flexibility, thus an own communication layer for the negotiation seems to be too much overhead.

For the description of each knowledge source (or Topic Agent interface) we decided to use RDF within XML. Our approach can also be compared to Service Data Objects (SDO) [18] which are describing information sources as abstract interfaces, but SDOs differ from our implementation because we do not need the strict abstraction level as well as we do not plan to write information directly back into the information source (in SEASALT this task is mainly carried out by the Case Factory). Further on SDOs return retrieval results as graphs that have to be queried using XMLPath which would be too much overhead in our approach. Other related approaches for the realisation of the knowledge map are the Business Execution Language (BEPL) [19], OWL(-S) [20], or WSDL [21], but all of them either did not fit our requirements to be easily maintainable by humans (like Knowledge Engineers) or had other objectives. For a more detailed evaluation of these approaches please see [9].

6 Evaluation and Future Work

A comparative evaluation of the coordination agent and its underlying knowledge map is difficult, since both are the solution to a rather specialised problem that, in this case, stems from the modularised nature of the SEASALT architecture. Also we think that a purely local evaluation with regard to performance and runtime would be of little value to fellow researchers. Because of this we chose to do a practical evaluation within our first application domain travel medicine. Our application partner's current best practice is the manual assembling of information leaflets, copy-pasting recurrent texts (like general information and warnings) from prepared templates and external

sources. The application partner has been compiling these information leaflets for several years and has in the meantime optimised the process as far as possible. Using this approach a trained medical practitioner needs about an hour to create a complete leaflet. First tests have shown that the docQuery system offers a significant time saving and takes a lot of repetitive tasks from the medical practitioner. Even when counterchecking every generated leaflet and, if necessary, adding corrections or additional information the process of composition of information leaflets is significantly accelerated using docQuery.

Concerning the actual implementation, in section 4.1 we defined several requirements and Table 1 lists these requirements in more detail. Since this is work in progress the evaluation table points out what the current state of our implementation is but also what we plan to do in the future.

Table 1: Evaluation of Implementation

Requirements	**Current Implementation**	**Future Work**
Access to local and remote knowledge sources	Possible	
Access via different access protocols	Yes (RMI (e:IAS), ODBC)	Web Services
Description of dependencies	Yes, in graph	Automatic detection of dependencies based on semantic markup
Knowledge Map Format	Yes, described using RDF and specialised namespaces	Complete description in RDF
Generation of request strategy	On demand, start and ending nodes have to be defined	Automatic recognition of start and ending nodes
Optimisation criteria	Yes, currently implemented: information *quality*, economic *cost*, *speed*, access *limits*	Evaluation of additional criteria
Combination of optimisation criteria	No, currently one criterion has to be chosen	(Weighted) combination of optimisation criteria
Definition of result set limitation	Yes (number of results, percentage, minimum similarity, “placement”)	Evaluation of additional limitations
Integration in a multi-agent-architecture	Yes using Jade [11]	Implementation of more flexible behaviour and parallelisation of requests using several Coordination Agents

Currently we use the Coordination Agent in the docQuery project. Due to its medical domain this project requires strict definitions and dependencies, so their implementation was our main focus in this first instantiation of SEASALT. For docQuery the Coordination Agent works satisfactorily, but in order to create a more

general and a more flexible architecture we will improve and extend the existing functionalities as pointed out in the evaluation table.

7 Conclusion and Outlook

This paper focused on the implementation of a Coordination Agent that can access distributed case bases, process and compile the retrieved information, and create individualised answers in a complex application domain. The Coordination Agent is the central component of the knowledge provision task within the SEASALT architecture that is based on the CoMES approach. The knowledge provision is realised using the Knowledge Line approach to coordinate different knowledge sources and provide a flexible framework for knowledge provision. In our docQuery application we handle seven different case bases with different case representations that have to be accessed in order to create complete information for a traveller. Further on, we plan to extend the knowledge sources with additional and redundant services aiming at better or more reliable results in case our case bases do not cover a certain request.

The Coordination Agent's main feature is the Knowledge Map containing abstract access methods for different kinds of knowledge sources as well as a graph-based representation of the knowledge sources themselves so we can explicitly define dependencies between knowledge sources as we claimed it in [3]. The adapted Dijkstra algorithm has proven to be a good choice to automatically calculate a request strategy. However, in more flexible application domains our algorithm might have to provide more features, such as the combination of optimisation criteria or the automatic detection of entry points into the graph, so our future work will especially focus on that area.

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. Bach, K., Reichle, M., Althoff, K.-D.: A Domain Independent System Architecture for Sharing Experience, In: Proceedings of LWA 2007, Workshop Wissens- und Erfahrungsmanagement, Martin-Luther-University Halle-Wittenberg, Germany, pp. 296–303
3. Althoff, K.-D., Reichle, M., Bach, K., Hanft, A., Newo, R.: Agent Based Maintenance for Modularised Case Bases in Collaborative Multi-Expert Systems. In Proceedings of the 12th UK Workshop on Case-Based Reasoning, December 2007 Cambridge, UK, pp. 7-18
4. empolis GmbH. Technical White Paper empolis:Information Access Suite. Technical report, empolis GmbH, September 2005.
5. 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)

6. 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)
7. 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)
8. Richter, M.M.: Introduction. In Lenz, M., Bartsch-Spörl, B., Burkhard, H.D., Wess, S., (Eds.): Case-Based Reasoning Technology – From Foundations to Applications. LNAI 1400. Springer-Verlag, Berlin (1998)
9. Reichle-Schmehl, A.: Entwurf und Implementierung eines Softwareagenten zur Koordination des dynamischen Retrievals auf verteilten, heterogenen Fallbasen. Bachelor's thesis, Institute of Computer Science, University of Hildesheim, September 2008.
10. Dijkstra, E. W.: A note on two problems in connexion with graphs. In: Numerische Mathematik. 1 (1959), pp. 269–271
11. Caire, G.: Developing multi-agent applications with JADE – Tutorial for beginners / TILAB. September 2007. – Technical Report. http://jade.tilab.com/doc/tutorials/JADE Programming-Tutorial-for-beginners.pdf ; last visited October 18th 2008
12. Ontañón, S. and Plaza, E.: An Argumentation-based Framework for Deliberation in Multi-Agent Systems. In: Rahwan, Iyad and Parsons, Simon and Reed, Chris (Eds) Argumentation in Multi-Agent Systems. LNCS, Vol.4946, p. 178-196. Springer (2008).
13. 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)
14. Redmond, M.: Distributed cases for case-based reasoning: Facilitating use of multiple cases. In: AAAI. (1990) pp. 304-309.
15. Briggs, B., Smyth, B.: Provenance, Trust, and Sharing in Peer-to-Peer Case-Based Web Search. In: Proceedings of the 9th European Conference on Case-Based Reasoning (ECCBR). Trier, Germany, LNAI 5239, pp. 89-103. Springer (2008).
16. Klusch, M., Sycara, K.: Brokering and Matchmaking for Coordination of Agent Societies: a survey. In: In Coordination of internet Agents: Models, Technologies, and Applications Springer-Verlag, London, (2001) pp.197-224.
17. Calisti, M., Faltings, B.: Constraint Satisfaction Techniques for Negotiating Agents, In: Proceedings AAMAS02 Workshop, July 2002, Bologna, Italy.
18. Adams, M., Andrei, C., Barack, R., Blohm, H., Boutard, C., Brodsky, S., Budinsky, F., Bünnig, S., Carey, M., Doughan, B., Grove, A., Halaseh, A., Harris, L., von Mersewsky, U., Moe, S., Nally, M., Preotiuc-Pietro, R., Rowley, M., Samson, S., Taylor, J., and Thiefaine, A.: Service Data Objects For Java Specification, Open Service Oriented Architecture collaboration. Technical Report, November 2006.
19. Jordan, D., Evdemon, J., Alves, A., Arkin, A., Askary,S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guízar, A., Kartha, N., Liu, C. K., Khalaf, R., König, D., Marin, M., Mehta, V., Thatte, S., van der Rijn, D., Yendluri, P., and Yiu, A.: Web Services Business Process Execution Language Version 2.0. Organization for the Advancement of Structured Information Standards. Technical Report. April 2007, http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, last visited on September 22nd 2008
20. McGuinness, D. L. and van Harmelen, F.: OWL Web Ontology Language. World Wide Web Consortium. Technical Report. February 2004, http://www.w3.org/TR/owl-features/, last visited on September 22nd 2008
21. Chinnici, R., Moreau, J. J., Ryman, A., and Weerawarana, S.: Web Services Description Language (WSDL) Version 2.0. World Wide Web Consortium W3C Recommendation, Technical Report. June 2007, http://www.w3.org/TR/wsdl20/ and http://www.w3.org/TR/wsdl20-adjuncts/, last visited on September 22nd 2008