SEASALTexp - an explanation-aware architecture for extracting and case-based processing of experiences from internet communities

SEASALT^{exp}— An Explanation-aware Architecture for Extracting and Case-Based Processing of Experiences from Internet Communities

Thomas Roth-Berghofer
Christian Sauer
School of Computing and Technology,
University of West London,
Ealing, London, United Kingdom
{firstname.lastname}@uwl.ac.uk

Klaus-Dieter Althoff
Kerstin Bach and **Régis Newo**
Competence Center CBR, German Research
Center for Artificial Intelligence DFKI GmbH
and University of Hildesheim, Germany
{firstname.lastname}@dfki.de

Abstract

This paper briefly describes SEASALT^{exp}, an extension of the application-independent SEASALT architecture (Sharing Experience using an Agent-based explanation-aware System Architecture LayouT), which offers knowledge acquisition from Internet communities, knowledge modularisation, and agent-based knowledge maintenance complemented with agent-based explanation facilities.

1 Introduction

The World Wide Web unquestionable contains lots of useful information. On the Web 2.0 quite a lot of information is personal, describing experience and ideas in blog or forum posts, on mailing lists or in wikis. The SEASALT approach provides a framework for sharing such experience using an agent-based system architecture layout [Reichle *et al.*, 2010]. It aims to develop a domain-independent and versatile architecture for intelligent information systems and subsequently specify and develop its individual components and finally implement it for different application scenarios. Its focus lies on domains that come with a surrounding community or can easily initiate an accompanying community.

The core methodology for SEASALT is case-based reasoning (CBR) [Aamodt and Plaza, 1994]. CBR is based on the hypothesis that similar problems have similar solutions. The cases in CBR systems are represented as sets of problem descriptions and their according solutions. Within SEASALT, the tasks to be executed are organised in layers according to the objectives they address: A variety of knowledge sources from the Web, tasks that concentrate on knowledge formalisation and knowledge provision, and a common knowledge representation.

SEASALT is very complex by its nature. For the user such questions can arise as about the knowledge and the knowledge sources used by a SEASALT instantiation (what-questions), about the justification of a suggestion (why-questions), or about the processing of a certain user query (how-questions). In the following we describe SEASALT^{exp}, an extension of SEASALT that provides SEASALT with explanation capabilities. The next section introduces the general explanation scenario and fits SEASALT in it. Section 3 briefly describes the SEASALT components and the new extensions of SEASALT^{exp}. The final section concludes the paper with a brief outlook on further research activities of our two research groups.

2 General explanation scenario

Explanations can be viewed as answers to questions, which are typically asked whenever one does not know about a certain concept, when a system (or, generally, a communication partner) behaves 'strangely', or results are unexpected. In a general explanation scenario (Figure 1) we distinguish three main communication participants [Roth-Berghofer and Richter, 2008]: the *user* who is corresponding with the software system via its user interface (UI), the *problem solver*, i.e., the actual tool or reasoning component, which provides the functionality for the original task of the software, and the *explainer*. Problem solver and explainer need to be tightly coupled in order to provide the necessary knowledge about the inner workings of the problem solver and its problem-solving processes.

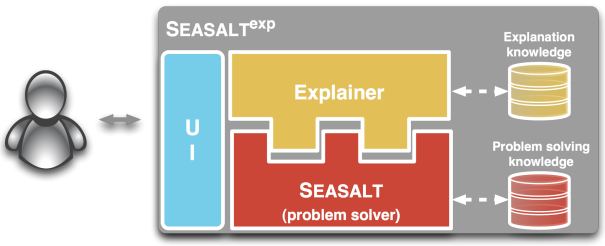

Figure 1: Communication participants in explanation scenario for SEASALT^{exp} with different knowledge sources (adapted from [Roth-Berghofer and Richter, 2008])

In rule-based expert systems looking at the rule trace was the main way of accessing the problem solver's actions. Given that the inferencing mechanism is fixed in those systems the trace was all the explainer needed. As soon as no trace is available the explainer has nothing to work with and, thus, cannot generate an accurate explanation of the reasoning process. The problem solver, then, needs to provide the necessary information. This is only possible if the software engineer developed such explanation capabilities into such a system.

In a SEASALT instantiation, agents work together in a complex fashion in order to provide an answer to a user query. This requires an equally intricate mechanism for observing and recording the decision processes in order to accurately describe to the user what, how, and why the system produced a certain result. As depicted in Figure 1 by the database symbols, problem solver and explainer each have their own knowledge-bases. SEASALT needs certain knowledge to answer user queries about a certain domain.

Only when the user has a question about a concept or action of SEASALT additional (explanation) knowledge is required. This is reflected in the extension of the knowledge models in the next section.

3 SEASALT^{exp}

SEASALT^{exp} provides an application-independent architecture that features knowledge acquisition from a web-community, knowledge modularisation, and agent-based knowledge maintenance. This section provides a high-level overview of its components as depicted in Figure 2. The individual components are grouped into layers according to their function in knowledge management. Knowledge is drawn from different *sources*, *formalised* by a knowledge engineer with the help of an apprentice agent and stored as *problem-solving and explanation knowledge*. SEASALT's knowledge provision is realised using the knowledge line approach [Bach *et al.*, 2008]. The knowledge line's basic idea is to modularise knowledge analogous to modularising software in the product line approach within software engineering [van der Linden *et al.*, 2007]. The *answer* procured by the *knowledge provision* layer then is further individualised and enhanced with explanatory information according to the needs of the enquirer.

Explanation-enhanced Interface The web-based interface allows the user to enter a semi-structured query by offering several text fields that can be filled with natural language. The user interface needs to account for different kinds of explanations, level of detail, and presentation style (see, e.g., [Atzmüller and Roth-Berghofer, 2010]).

Case Factory For each *Topic Agent* a *Case Factory* takes care of its knowledge maintenance. It comprises a number of agents that each carries out a simple maintenance task such as adding new cases, preserving consistency, or generalising redundant cases. As the changes of the system's knowledge changes the behaviour of the system and, thus, may trigger questions of the user, the changes need to be monitored. An *Observer Agent* takes care of this task.

Knowledge Line The Knowledge Line provides a facility for modularising the knowledge held in a complex application domain. It comprises the *Coordination Agent*, the *Explanation Agent*, and several *Topic Agents*.

Intelligent Interface The *Intelligent Interface* supports the *Knowledge Engineer* in the task of formalising relevant posts. It offers input assistance, a controlled vocabulary, and access to the *Problem-solving Knowledge*.

Knowledge Engineer The *Knowledge Engineer* is the link between a *Community* and its *Topic Agents*. He or she receives posts which the *Collectors* deem relevant regarding one of the fields, represented by the *Topic Agents*, and formalises them for insertion in the *Topic Agents'* knowledge-bases using the *Intelligent Interface* and being supported by the *Apprentice Agent*. Some of the interactions of the *Knowledge Engineer* with the various agents are relevant to certain questions the user might ask. An *Observation Agent* takes care of recording them for later use by the *Explanation Agent*.

Apprentice Agent The *Apprentice Agent* supports the *Knowledge Engineer* in formalising relevant posts for insertion in the *Topic Agents'* knowledge bases. It is trained by the *Knowledge Engineer* with community posts and their formalisation. In [Bach *et al.*, 2010], we describe the Knowledge Extraction Workbench (KEWo), an approach for knowledge extraction for Case-Based Reasoning systems. KEWo is an example implementation of an *Apprentice Agent*.

Community The *Community* is the source of the knowledge held by the *Topic Agents*. The *Community* uses the *Intelligent Platform* to communicate and share experiences concerning the respective application domain. Their posts are crawled by *the Collectors*.

Collector Agent *Collectors* crawl the *Intelligent Platform*. Each collector is associated to an individual *Topic Agent* and collects community posts that include information that is relevant for insertion in their respective *Topic Agent's* knowledge base. Recording the availability of knowledge sources and the availability or absence of answers to Topic Agent's queries is among the tasks of an *Observation Agent* here.

Intelligent Platform The *Intelligent Platform* offers a communication and collaboration platform to the *Community*. It is enhanced with intelligent agents that offer content-based services such as the identification of experts, similar discussion topics, etc.

Problem-solving and explanation knowledge This layer holds knowledge models such as rules, vocabulary, ontologies, taxonomies, similarity measures, etc. They are input to all agents, the *Intelligent Platform*, and the *Intelligent Interface*. The observations collected by the *Observer Agents* are input to the *Explanation Agent* and the *Explanation Factory*. Each of the knowledge models are enhanced by knowledge that is only relevant to explanations such as explanations of rules or constraints.

Topic Agent Each *Topic Agent* serves as an expert for a particular aspect or field of the application domain. A *Topic Agent* can be any kind of information system or service including CBR systems, databases, web services, etc.

Coordination Agent The *Coordination Agent* receives the natural language query, analyses it and subsequently queries respective *Topic Agents* using incremental reasoning, i.e., by using one agent's output as the next agent's input. To do so it uses a *Knowledge Map* which lists all *Topic Agents*, the information they offer, and their dependencies that can be navigated like a graph. Finally the *Coordination Agent* uses the query results and prefabricated templates to compose an answer to be given to the user.

Explanation Agent On the abstract level of the communication scenario the *Coordination Agent* stands as main component for the problem solver, i.e., SEASALT, and the *Explanation Agent* for the explainer. The *Explanation Agent* knows about the application domain and the tasks and processes of each topic as well as the *Apprentice Agent* and the *Coordination Agent* via the *Problem-solving and Explanation Knowledge*.

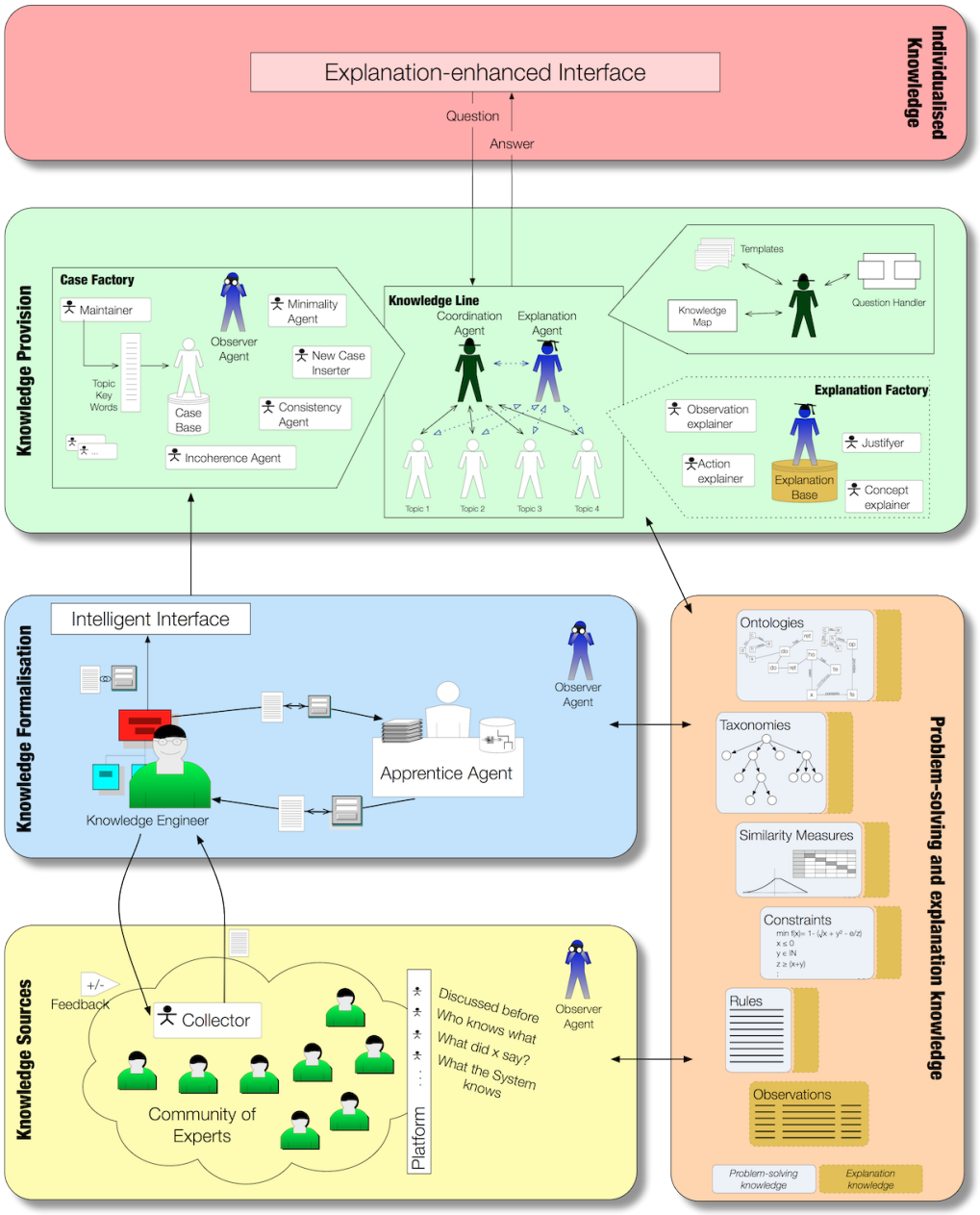

Figure 2: SEASALT^{exp} overview

Observer Agent The *Observer Agents* watch and record interactions between agents as well as between the *Knowledge Engineer* and the *Apprentice Agent*. The *Explanation Agent* makes use of the recorded observations when the user has questions about (unexpected) query results.

Explanation Factory The *Explanation Factory* makes use of the available *Problem-solving and Explanation Knowledge*. A number of agents carry out explanation tasks such as providing concept explanations, action explanations, and justifications. The *Observations*, recorded in the *Problem-solving and Explanation Knowledge* are an important input to them, in general, and to the observation explainer, in particular.

4 Conclusion and Outlook

SEASALT^{exp} is an explanation-enabled variation of the application-independent SEASALT multi-agent approach. In this paper, we gave a high-level overview of the components and their interplay. The SEASALT architecture offers several features, namely knowledge acquisition from Web 2.0 communities, modularised knowledge storage and processing and agent-based knowledge maintenance. SEASALT^{exp} adds general explanation capabilities. With the integration of *KEWo* into the open-source CBR tool *myCBR 3*[1] we have added particular explanation capabilities to an *Apprentice Agent* in a SEASALT instantiation [Sauer and Roth-Berghofer, 2011].

Another area of research that we currently look into are trust and provenance of information. SEASALT incorporates information from a large number of sources and we are currently looking into methods for making the source of the individual pieces of information more transparent to users and, thus, improve the system's acceptance and trustworthiness.

[1] More on *myCBR 3*: `http://mycbr-project.net`

References

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

[Atzmüller and Roth-Berghofer, 2010] Martin Atzmüller and Thomas Roth-Berghofer. Ready for the MACE? The Mining and Analysis Continuum of Explaining uncovered. Research Report RR-10-02, Deutsches Forschungszentrum für Künstliche Intelligenz DFKI GmbH, 2010. ISSN 0946-008X.

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

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

[Reichle *et al.*, 2010] Meike Reichle, Kerstin Bach, and Klaus-Dieter Althoff. Knowledge engineering within the application independent architecture SEASALT. In Joachim Baumeister and Grzegorz J. Nalepa, editors, *Int. J. Knowledge Engineering and Data Mining*, pages 202–215. Inderscience Publishers, 2010.

[Roth-Berghofer and Richter, 2008] Thomas R. Roth-Berghofer and Michael M. Richter. On explanation. *Künstliche Intelligenz*, 22(2):5–7, May 2008.

[Sauer and Roth-Berghofer, 2011] Christian Severin Sauer and Thomas Roth-Berghofer. Web community knowledge extraction for myCBR 3. In Max Bramer, Miltos Petridis, and Adrian Hopgood, editors, *Research and Development in Intelligent Systems XXVIII. Proceedings of AI-2011. The Thirtyfirst SGAI International Conference on Artificial Intelligence*. Springer, 2011. To appear.

[van der Linden *et al.*, 2007] Frank van der Linden, Klaus Schmid, and Eelco Rommes. *Software Product Lines in Action - The Best Industrial Practice in Product Line Engineering*. Springer, Berlin, Heidelberg, Paris, 2007.