Multi-agent, case-based configuration of custom-built racing cars

Carsten Wenzel[2], Pascal Reuss[1 2], Karsten Rose[3], and Klaus-Dieter Althoff[1 2]

[1] German Research Center for Artificial Intelligence
Kaiserslautern, Germany
`http://www.dfki.de`

[2] Institute of Computer Science, Intelligent Information Systems Lab
University of Hildesheim, Germany
`http://www.uni-hildesheim.de`

[3] KER INNOVATEC GMBH, Nordstemmen, Germany

Abstract. The assembly of a custom-built automobile can be challenging, because of the multitude of car components that can be used and their configuration possibilities among one another. Several components are combined to create a car component, for example six parts are used to create a rear axle and several constraints have to be considered for a successful configuration. In this paper we present the work done within a masters thesis [19], namely a prototype of a multi-agent system for case-based configuration of car components, especially for creating new components by using preexisting components.

Keywords: *Case-Based Reasoning, Configuration, Multi-Agent-Systems*

1 Introduction

In the domain of car component configuration preexisting components are used, For this task a large number of individual components has to be considered. Components of the same kind are similar to one another with respect to their functionality but differ in one ore more attributes. Further, while assembling components to create a new component, there are constraints which have to be satisfied. For example, if someone wants to change the rim of the car to get a racy look, there are thousands of possible rims, but not everyone fits. This depends on several constraints of the car, like its wheel fixture, engine performance, ground clearance and further more. Additionally, there could be possibilities to adapt parts of the car to get other rims fitting, like using adapter plates to change the size of wheel fixture. Even more complex it is to convert a normal car into a racing car, because more than one part of the car has to be changed.

In this paper we present a prototype of a multi-agent system based on the SEASALT architecture, which can help to construct new components of preexisting car components that fit given requirements by using case-based reasoning (CBR). The prototype was developed with KER-INNOVATEC, a specialist for tuning AUDI cars. In the following we will introduce KER-INNOVATEC and the specific problem the prototype deals with. In Section 2 we describe the SEASALT architecture and the realized components within our prototype. Section 3 contains related work about case-based and rule-based configuration. In Section 4 we describe the configuration process and implementation of our multi-agent system in more detail and give a short example of a configuration. Section 5 shows our evaluation setup and the results and Section 6 gives a short summary and an outlook on future work.

1.1 KER INNOVATEC

In the early 1990s KER INNOVATEC [12] began with the sale of spare parts. Almost at the same time they started the production of fiber-glass reinforced plastic parts (GRP). In 1995 KER INNOVATEC built their first Sport Quattro with a short wheelbase. At the end of the 1990s they started with the development and production of stainless steel exhaust systems for AUDI cars. More and more customers wanted a custom design directly to their vehicle, for example Ferraries, customized racing cars, among other things for 24-hour race. Starting in 2000 the focus was on the body parts made of plastic. In 2004 KER INNOVATEC started the development of other vehicles. Partial two vehicles were simultaneously developed and built. In 2008 matured the decision to build a proper successor to the Sport Quattro. KER INNOVATEC was also a pioneer and shortened as the first one a Type 89 Coupe Quattro body. In spring 2013 the first vehicle was ready. Together with their customers KER INNOVATEC always incur new projects, some of them being very sophisticated custom-made projects with special applications and requirements. From the first meeting to delivery to the customer fun, enthusiasm, and precision work accompany one another. KER INNOVATEC now manufactures car parts for dealers, race teams, and other interested buyers according to their specifications. KER INNOVATEC also builds special tools in collaboration with other partners.

A problem which can appear is that by doing many transformations from the original car, like changing the complete engine, the whole backward drive section (chain) has to be redesigned. This problem was chosen as the first component our prototype multi-agent system should deal with. The backward drive section from four-wheel drive cars consists of six parts where at least each part has various attributes for its characterization. These are the gearbox, the rear axle differential, the drive-shaft, the wheel hub, the wheel bearing, and the suspension strut. The gearbox is the connection between the engine of the car and the backward drive section, the counterpart is the suspension strut, it is the least point of the backward drive section with connection to the vehicle body. The other parts connect these two units. One part depends on attributes given from the car and also from other parts of the backward drive section. In the past, the approach has been proven to be the best way to reach suitable results while constructing a new backward drive section to begin with the gearbox and the suspension strut. To get this done, in the past always an extensive search on the web for individual components and configuration possibilities began. About the last years also satisfying, customized solutions were stored to reuse them. With our prototype we want to improve the way of searching for suitable solutions. To get this done we decided to implement a case-based configuration system based on the SEASALT architecture.

2 SEASALT architecture

Our approach is based on an architecture named SEASALT (Sharing Experience using an Agent-based System Architecture LayouT), which describes a domain independent architecture for extracting, analyzing, sharing, and providing experiences [4]. Until now the main scope of SEASALT was extracting knowledge from textual knowledge sources, for example social communities such as forums, wikis, or discussions on social platforms. SEASALT aims at providing a general architecture independent from specific technological restrictions. The architecture is geared to real world scenarios where certain people are experts in special domains and the knowledge of more than one expert as well as the composition of a combined solution are required in order to solve a complex problem. Based on the Collaborative Multi-Expert-Systems (CoMES) approach [1], artificial intelligence technologies are used to identify relevant information, process the experiences and provide them via an interface using a collaborating multi-agent architecture. Modularizing knowledge into snippets (pieces of knowledge within one topic) allows the compilation of comprehensive solutions, rather than restoring complete episodic cases. Further reusing partial case information in form of snippets is possible, since each module will cover a particular topic. The components of the SEASALT architecture are presented in the following.

Knowledge sources
Many real-life application domains need to draw information from numerous knowledge sources in order to keep up to date. Beyond traditional knowledge sources such as databases and static web pages SEASALT also considers Web 2.0 platforms. The SEASALT architecture is especially suited for the acquisition, handling, and provision of experiential knowledge as it is, for example, provided by communities of practice and represented within Web 2.0 platforms [17].

Knowledge formalization
In order for the extracted knowledge to be easily usable within the Knowledge Line the web contributions have to be formalized from their textual representation into a more modular, structured representation. This task is mainly carried out by the Knowledge Engineer. The role of the Knowledge Engineer can be carried out human experts, software systems, or a combination of both. The Knowledge Engineer is the link between the Knowledge Sources and the Topic Agents. The Intelligent Interface serves as the Knowledge Engineer's case authoring work bench for formalizing textual knowledge into structured CBR cases. It has been developed analogous to [3] and offers a graphical user interface that presents options for searching, browsing and editing cases and a controlled vocabulary [4].

Knowledge provision
SEASALT's knowledge provision is realized using the Knowledge Line approach. The Knowledge Line's basic idea is a modularization of knowledge analogous to the modularization of software in the Product Line approach within software engineering [14]. Within the SEASALT architecture this knowledge modularization happens with regard to individual topics that are represented within the respective knowledge domain. These topics are represented by Topic Agents. According to the SEASALT architecture the Topic Agents can be any kind of information system or service including CBR systems, databases, web services, or other kinds of machine accessible knowledge stores. Additionally the Topic Agents' CBR systems are extended with Case Factories, which take care of the individual agents' case maintenance. The Topic Agents are orchestrated by a central Coordination Agent. The Coordination Agent receives a semi-structured natural language query from the user, analyzes it using a rule-based question handler and subsequently queries the respective Topic Agents using incremental reasoning, that is using one agent's output as the next agent's input. In doing so the Coordination Agent's course of queries resembles the approach of a human amateur trying to answer a complex question. The comprehensive and general knowledge needed in order to carry out this incremental reasoning process is represented within the so called Knowledge Map, which provides formal representations of all Topic Agents and possible output / input connections encoded in a graph-like structure. Finally the Coordination Agent uses the query results and prefabricated templates to compose the information to be given to the user [2].

Knowledge representation
Since the miscellaneous agents operating on the Knowledge Sources, the Knowledge Engineer's tools and the CBR systems of the individual Topic Agents deal with the same knowledge domain(s), it makes sense to join their underlying knowledge models. This does not only greatly facilitate knowledge model maintenance but also allows for an easier interoperability between the individual components.

Individualized knowledge
The user interacts with the SEASALT-based application via a web-based interface. The web based interface offers a semi-structured input in the form of different text fields used for entering information the end user can easily provide.

2.1 Instantiation of the SEASALT architecture

This subsection describes the instantiation of the components of the SEASALT architecture within our multi-agent system. Not all components are instantiated in the prototype yet, but could be instantiated in future developments.

Knowledge sources
The knowledge to define case structures, case content, and rules is provided by the experts from KER-INNOVATEC. This knowledge was mainly in form of databases or excel sheets and the experience of the technicians. While the

knowledge from database and excel sheets was already structured, the experience had to be explicitly written down during interviews with the experts.

Knowledge formalization
In the prototype of our multi-agent system, the part of the Knowledge Engineer is carried out by the experts of KER-INNOVATEC and the developers of the multi-agent system. Because the formalization was done by hand, an implementation of an apprentice agent was also out of scope of our prototype.

Knowledge representation
In our multi-agent system different types of knowledge are required, like structured case, similarity measures, or rules. These different kinds of representations are used by the specialized types of the topic agents. We use the *myCBR* workbench[5] to create the case structure and the similarities of the CBR component for the different topic agents. The tool *Drools* [7] was used to model the rules.

Knowledge provision
Within the actual problem of redesigning a backward drive section, the different topics were identified as the several parts of the drive section. Each topic is represented by a separate topic agent. Actual two different types of topic agents are implemented, one type operates with cased based reasoning to answer requests, the other one also operates with case-base reasoning supplemented with rule-based reasoning.

While answering a request to the system, these topic agents search for suiting parts in their specific topic. A sequential retrieval is used to find a configuration and the output of one topic agent is used as a part of the request for the following topic agent. To handle this request to the topic agents, another type of agent is used, called coordination agent. This agent has several tasks. The first task is to receive a request from a third type of agent, called communication agent. It also takes care of handling the requests to the topic agents and summarizes the individual replies to create an overall solution to the request for the user. This overall solution is sent back to the communication agent. For splitting the query, enriching the query during the retrieval, and generating an overall solution the coordination agent requires the knowledge about the problem decomposition, the sequence of required topic agents, and the combination possibilities of the individual solutions. This knowledge is stored in a so-called Knowledge Map. The Knowledge Map provides a formal representation of all topic agents and possible output and input connections encoded in a graph-like structure. [4]. The task of the communication agent is to handle the communication between the user-interface and the multi-agent system.

Individualized knowledge
For user interactions with the system, a user-interface has to be implemented. Different possibilities for this interface exist, like a web-based interfaces, a smartphone interface or a desktop application. This interfaces communicates with the multi-agent system via an XML document. This was chosen to get the ability to handle different kinds of interfaces. In our prototype we decided to implement a desktop application in JavaFX. The application offers the users the possibility

to enter their individual requirements and it also provides different possibilities to show the answer of the system. For further developments, like adding new attributes to a component, the data structure of the interface is dynamical and an update functionality is implemented, which adapts the user interface based on the knowledge changes inside the CBR system. The answer of the multi-agent system can be shown in three different views. The first one is a list view sorted by the similarity of retrieved configurations. By this view a navigation to the other two views is possible. The second view shows the selected configuration with its components with the corresponding attributes. The third view presents the retrieval network as a graph structure created by the system.

3 Related work

Hybrid systems using different kinds of knowledge have been developed in the last years since CBR was developed as alternative to rule-based reasoning (RBR). Also other kinds of reasoning take part in these hybrid systems. One advantage of hybrid systems is, that different reasoners could be used and the strength of these reasoners could be combined. In this section we give a short overview of research in the knowledge-based configuration field of the last years. We describe hybrid configuration systems and how they were used to solve different kinds of problems in the domains of configuration or design as well as other configuration related research.

Guenther and Kuehn gave an overview of knowledge-based configuration approaches including case-based configuration. There are two types of approaches for case knowledge selection. The first type retrieves a complete configuration and adapts this configuration if possible. The second type retrieves parts of the configuration in sequential retrievals and builds the complete configuration based on the individual parts [9]. In our prototype we use a combined approach of these two types: we retrieve parts of the configuration from individual case bases and combine them to a complete configuration and the individual parts as well as the complete configuration may be adapted if possible. Hotz an Krebs focuses in their work on state of the art for knowledge-based configuration on the challenges to be met. They focus mainly on software configuration in the context of product lines, but the challenges to be met are similar in other domains, like explainability, integration of configuration into business processes, and knowledge elicitation [11]. During the development of our prototype we had to face these challenges, too. Eliciting configuration knowledge from technicians and transform it to an explicit form was very difficult. Also the integration of the configuration into the daily business process was a challenge, because the prototype should be used in daily workflows.

In his PhD thesis, Thorsten Krebs introduces a framework for supporting the evolution of configurable products. This framework should help to manage the configuration requirements and dependencies when the specifications of a product or a product family changes over time [13]. Some of his approaches may be useful in other domains, especially for custom-built products, where the customer wishes and therefore the product specifications may change during the production process.

Already in the early 1990's with Clavier a CBR system was developed which satisfied industrial requirements in the domain of configuration. Clavier was developed to assist in determining efficient loads of composite material parts to be cured in an autoclave [10]. One of the main reasons for using the CBR methodology in Clavier was that CBR can deal with a small initial case base and also allows to expand and refine the knowledge in the knowledge base over time. As retrieval mechanism in Clavier the CBR system has two inputs, the case memory of previously run autoclave loads and the list of parts that need to be manufactured. As a result Clavier presents several possible loads in a ranked order. If no exact match was found the user has the possibility to adapt one of the presented loads. This adaption is checked by Clavier for its consistency. If the adaption is valid the system generates the load of the autoclave. After the load has been in the autoclave, the operator has to tell Clavier if the result of the load was successful, or not. This knowledge is used in Clavier to generate new cases for the case base. In addition this knowledge can be used for validation of adapted solutions. An additional reason for the selection of CBR in Clavier was that even experts in this domain have problems to define a successful load of the autoclave if they have to handle a new situation. These experts need to have experience with this new situation before they could reason about it. Therefore the experts were not able to define new rules without corresponding to a specific load they had previously cured in the autoclave. [10]

In the late 1990s with CAMPER a hybrid system was developed using CBR and RBR to meet multiple design constraints in the domain of nutritional menu planning. The case base of the CBR component consists only of positive menus suiting to individual requirements. The retrieval mechanism tests the actual requirements against each of the menus stored in the case base. If a requirement is not met, a penalty score is assigned on how difficult it would be to bring the case into compliance. The menu with the lowest penalty score is chosen as the case passed to the rule engine. The rule engine in CAMPER is responsible for the adaption of the selected menu. [15]

In our prototype CBR is used to search in the different topics for car parts suiting the actual requirements and RBR is used for adaption of car parts and for constraint checking. In difference to the described systems above, our prototype uses CBR not to retrieve a full or partial configuration, but to retrieve the individual parts of the component. The successfully approved configurations can be stored again. These complete configurations are not used yet, which is of course one of the next improvement steps. The rule based component is used to adapt a part to fit a given constraint, but not every part can be adapted, so it is possible to find no suitable configuration for a query.

4 Configuration task and realization of the prototype

This section describes the addressed configuration task and the realization of this task within our prototype. While assembling a custom-built car several components have to be reconfigured to fit the given constraints and customer visions. Because of the high effort for new developments, existing parts are used and adapted to build new components. For each part several different versions exist, varying in some values like diameter or circumference. One challenge is to decide whether a certain part can be used for a configuration, or not. Adapting a part that is not usable with its original characteristic, is another challenge. KER INNOVATEC has gathered much experience in these challenges. In addition, not only constraints for parts have to be considered, but constraints for the whole component, too. For example a constraint for a single part may be the diameter fitting into a certain space. If the space has a diameter of 110 millimeters, then the part has to have a diameter that is less than 110 millimeters. A constraint for a whole component is the total length, which depends on the car's body. While the total length is fix, the length of the individual parts may vary. The task for our prototype is the configuration of the rear axle. A rear axle consists of six parts: gear box, HA differential, drive shaft, wheel hub, wheel bearing, and suspension strut. Figure 1 shows all six parts from left to right. For each part several different instances exist, which can be combined to a complete rear axle. During the configuration process the part and component constraints have to be met.

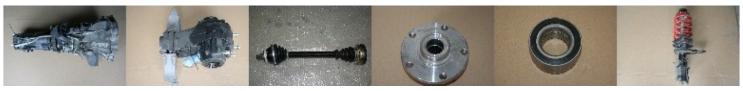
Fig. 1. The six parts of a rear axle

In cooperation with the experts from KER INNOVATEC three main goals for the application were formulated: the system should support the search for individual parts, the system should support the configuration of a whole component and the solutions of the system have to be represented in a form that can be interpreted by car engineers.

The knowledge required for the configuration of the rear axle was converted into so-called snippets. A snippet is a discrete information that is described independent from other snippets. It is combinable with other snippets and has knowledge about its validity in context of an individual or overall solution [16]. For our prototype each part of the rear axle is represented as a snippet. This way each snippet is interpreted as a sub-domain of the knowledge required for the configuration task. For each sub-domain a CBR system is realized with its own case structure and similarity measures. In addition rules are modeled to describe the dependencies between the sub-domains and for validating the solutions.

As an example we want to describe the snippet of the wheel bearing. The wheel bearing is specified via four attributes: the car type, the width, the outer diameter, and the inner diameter. In the CBR system we implemented two different kinds of knowledge representation for these attributes to define the local similarity measures. For the car type a taxonomical representation was chosen. The width, the outer diameter, and the inner diameter were defined trough numeric similarity functions. In addition an amalgam function was defined, for aggregating the local similarity measures to a global similarity measure. The wheel bearing has to suit to the suspension strut and the wheel hub. Therefore the outer diameter of the wheel bearing has to have the same size as the inner diameter of the application point at the suspension strut. Furthermore the inner diameter of the wheel bearing has to have the same size as the outer diameter of the wheel hub. To check these dependencies we implemented rules. If the CBR system suggests a part not suiting these rules, it is refused.

For a configuration of a component two entry points exist. The first entry point is the gear box and the second point is the suspension strut. That means, a configuration can be based on one of these entry points. If the entry point is the gear box we call the process a forward configuration, because usually a axle configuration starts with the gear box. If the entry point is the suspension strut we call the process a backward configuration.

Our prototypical multi-agent system consists of ten software agents. A communication agent is responsible for the communication with the user interface. The tasks of this agent are to read the query and sending it to the coordination agent as well as getting the solution and sending it back to the user interface. The coordination agent plays a central role in the configuration process. It has several responsibilities: examining the query, identifying the required sequence of knowledge sources and sending the query to them, combining individual solutions from each knowledge source to an overall solution and sending the overall solution to the communication agent. The coordination agent uses a so-called Knowledge Map to identify the required sequence. The idea of the Knowledge Map is from [8]. The idea was adapted to multi-agent systems by [6] and extended by [18]. The use of the Knowledge Map in the retrieval process is described later on. Checking the overall solution against the configuration constraints of the component has been delegated to an additional agent. This controller agent gets an overall solution from the coordination agent, checks the constraints, and sends the result back to the coordination agent. If an overall solution is rated as false, the solution is dismissed for the actual request. For each of the six mentioned sub-domains a topic agents with an underlying CBR system exists. A topic agent is responsible for retrieving the most similar cases to a given query. The six topic agents are processed in sequence not in parallel, because the query for the next topic agent has to be enriched with information from the retrieved cases. For example, the first topic agents retrieves a certain gear box. This gear box has a certain disc diameter and gear ratio. The values of these attributes have to be considered by the next topic agent, because they define constraints for the next part. Each topic agent has a rule component to check the retrieved cases for the given constraints. If a constraint is not met, the retrieved case is dismissed for this request. At last an update agent is responsible for dynamically updating the user interface based on the case structure of the CBR systems.

The retrieval process for configuring a component starts at an entry point, depending on the query. The coordination agent uses the Knowledge Map to identify the retrieval sequence. In our prototype two sequences exist, one for forward configuration and one for backward configuration. Within a sequence the order of the knowledge sources (topic agents) is fixed, due to the characteristics of the axle configuration. The Knowledge Map can be represented as directed graph and is implemented in RDF format. The Knowledge Map contains knowledge about the individual topic agents and their dependencies and connections between one another. The connection information is used to enrich the query with the required attributes from the retrieved solutions. There are different entries for the specification of the topic agent like its topic, the type of the topic agent, the similarity threshold to be met and the linking information between the topic agents.

After the required retrieval sequence is identified, the query is sent to the first topic agent. Based on the number of retrieved parts, for example gear boxes, the number of requests for the next topic agent in the sequence is determined. For every retrieved gear box the query has to be enriched and sent to the next topic agent. The same procedure is done for the following topic agents in the sequence. The number of additional queries for each topic agent can grow exponentially. For example, the first topic agent retrieves 12 results. The second topic agent gets 12 queries and retrieves 14 results for each query. Hence, the third topic agent already has to process 168 queries. To limit this exponential growth, the number of retrieved cases was limited. This way the retrieval process can be done with reduced effort and time, but not all possible configurations are retrieved. When all topic agents of the retrieval sequence were requested and the found configurations are assembled, the controller agent checks the configuration against the constraints. The approved configurations are sent to the user interface and represented as a graph structure. Each configuration can be highlighted and for each part a detailed view is available, which contains the attributes and their values.

5 Evaluation

In this section we describe the evaluation of the results generated through our prototype. The prototype was evaluated by the experts of KER INNOVATEC. The test data we used in the evaluation consists of at least five instances of every component, three of these single components are part of a preexisting backward drive section. Further one component can replace a component of the preexisting backward drive sections and one component that is not suitable with the other components. Additionally several random components were part of the data used in this evaluation to check the results generated by our prototype. First we evaluate if the prototype is able to reconstruct the three preexisting backward drive sections, if the requirements are suiting for these. In a second step the additional possibilities were evaluated. At least the rule-based component was evaluated.

The evaluation leads to the result that under suiting circumstances the system was able to identify the preexisting backward drive sections. With the limited number of different data we deal with, the prototype generates round about 55 different path trough the retrieval network in addition to the identified preexisting backward drive sections. These paths correspond to possible backward drive section. This is a large number of paths for our limited data and may cause a problem when scaling up the system with more data. On average 60 percent of the retrieved configurations are rated correct and useful by the experts of KER INNOVATEC. The results for the individual queries are displayed in the following figure.

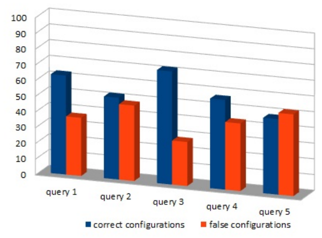

Fig. 2. Evaluation results for the five test queries

6 Summary and Outlook

With the prototype of our multi-agent system we were able to demonstrate the use of the SEASALT architecture in the domain of configuration components of a car with using preexisting parts. Even without implementing every component suggested in the SEASALT architecture, the prototype supplies confident results. In further developments more parts have to be stored to test the stability of the results and to increase the number and the quality of possible results the system could compile. Additionally different input options should be implemented and the similarity measures also have to be adapted and improved. While looking forward to the rule-based component, further rules for adaption have to be implemented and the ability to work with conflicting rules has to be improved.

Also the development of the different kinds of agents, like agents for knowledge acquisition or knowledge formalization, which are part in the SEASALT architecture, should improve the actual prototype. Thus, the very time-consuming work of searching for further knowledge in the web as well as the knowledge formalization should become more effective.

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 knowledge-product-lines with case factories and distributed learning systems. In: Baumeister, J., Seipel, D. (eds.) KESE @ KI 2007. Osnabrück (Sep 2007)
2. 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)
3. Bach, K.: Domainmodelling in textual case-based reasoning (in german). Master's thesis, University of Hildesheim (2007)
4. Bach, K.: Knowledge Acquisition for Case-Based Reasoning Systems. Ph.D. thesis, University of Hildesheim (2013), dr. Hut Verlag Mnchen
5. Bach, K., Althoff, K.D.: Developing case-based reasoning applications using mycbr 3. In: Case-Based Reasoning in Research and Development, Proceedings of the 20th International Conference on Case-Based Reasoning (2012)
6. Bach, K., Reichle, M., Reichle-Schmehl, A., Althoff, K.D.: Implementing a coordination agent for modularised case bases. In: Proceedings of the 13th UK Workschop on Case-Based Reasoning. pp. 1–12 (2008)
7. Balo, M.: Drools JBoss Rules 5.X Developer's Guide. Packt Publishing Birmingham, UK (2013)
8. Davenport, T.H., Prusak, L.: Working Knowledge: How Organizations Manage What they Know. Havard Business School Press (2000)
9. Guenther, A., Kuehn, C.: Knowledge-based configuration - survey and future directions -. In: Lecture Notes in Computer Science. vol. 1570, pp. 47–66 (1999)
10. Hinkle, D., Toomey, C.N.: Clavier: Applying case-based reasoning to composite part fabrication. In: Proceedings of the Innovative Application of Atrificial Intelligence Conference 1994 (1994)
11. Hotz, L., Krebs, T.: Configuration state of the art and challenges. 17. Workshop Planning, Scheduling and Configuration, Design (PuK) (2003)
12. Ker-Innovatec: Ker-innovatec website, `http://www.ker-innovatec.de/en/`
13. Krebs, T.: A Knowledge Management Framework That SSupport Evolution of Configurable Products. Ph.D. thesis, University of Hamburg, Institut of Computer Science (2008)
14. van der Linden, F., Schmid, K., Rommes, E.: Software Product Lines in Action - The Best Industrial Practice in Product Line Engineering. Springer Verlag Berlin (2007)
15. Marling, C., Petot, G., Sterling, L.: Integrating case-based reasoning and rule-based reasoning to meet multiple design constraints. Computational Intelligence 15, 308–332 (1999)
16. Marter, S.: Case-Based Coordination AAgent - Knowledge modularization and knowledge composition for decentralized, heterogenous case bases (in german). Master's thesis, University of Hildesheim (2011)
17. Plaza, E.: Semantics and experience in the future web. In: Proceedings of the 9th European conference on Advances in Case-Based Reasoning (2008)
18. Reuss, P.: Concept and implementation of a Knowledge Line - retrieval strategies for modularized, homogeneous topic agents within a multi-agent-system (in German). Master's thesis, University of Hildesheim (2012)
19. Wenzel, C.: Case-based configuration of custom-built rear axles for rallye cars. Master's thesis, University of Hildesheim (2014)