A Case-Based Reasoning Approach for Providing Machine Diagnosis from Service Reports

Kerstin Bach[1,2], Klaus-Dieter Althoff[1,2], Régis Newo[1,2], and Armin Stahl[1]

[1] Competence Center Case-Based Reasoning
German Research Center for Artificial Intelligence (DFKI) GmbH
Trippstadter Strasse 122, 67663 Kaiserslautern, Germany
firstname.surname@dfki.de
[2] University of Hildesheim
Institute of Computer Science - Intelligent Information Systems Lab
Marienburger Platz 22, 31141 Hildesheim, Germany
lastname@iis.uni-hildesheim.de
http://www.dfki.de/web/competence/cccbr/

Abstract. This paper presents a case-based reasoning system that has been applied in a machine diagnosis customer support scenario. Complex machine problems are solved by sharing machine engineers' experiences among technicians. Within our approach we made use of existing service reports, extracted machine diagnosis information and created a case base out it that provides solutions faster and more efficient than the traditional approach. The problem solving knowledge base is a data set that has been collected over about five years for quality assurance purposes and we explain how existing data can be used to build a case-based reasoning system by creating a vocabulary, developing similarity measures and populating cases using information extraction techniques.

Keywords: Case-based reasoning, machine diagnosis.

1 Introduction

Making use of available data in companies is one of today's major challenges. We collect a lot of data, have various information systems, and are able to track and record many actions people or machines perform. However, getting the best value out of our data might be easy for an expert as long as there are not too many data sets. Transferring this expertise into machine logic is still a challenging task. Within this paper, we describe how we pick up this challenge and explain how case-based reasoning can be applied to existing data within a company in order to create a knowledge-based system that makes work more effective.

Case-Based Reasoning (CBR) is an approach to solve new problems by adapting solutions of similar past problems [1]. CBR is also viewed as a methodology for building knowledge-based systems. In its core functioning CBR is based on experience, often called case-specific knowledge (cases), usually in the form of problem-solution pairs that are stored in a database of cases (case base) along

A. Ram and N. Wiratunga (Eds.): ICCBR 2011, LNAI 6880, pp. 363–377, 2011.

with continuative general knowledge. The latter is used for defining a similarity notion between problem descriptions (similarity measures), which supports partial (non-exact) matching in the case base, and case adaptation (adaptation knowledge).

To develop a CBR system its basic knowledge containers vocabulary, case base, similarity measures, and adaptation knowledge have to be filled [14]. Based on the defined vocabulary the cases have to be described as well as the general knowledge underlying similarity assessment and case adaptation. It is an important degree of freedom of CBR systems that a lack of knowledge in one of its containers can be compensated by using additional knowledge in the other knowledge containers. For instance, having a large number of cases available can compensate for a less precise vocabulary, rather rough similarity measures, and/or a lack of adaptation knowledge.

Especially when dealing with existing data and given representations one usually has a huge amount of cases available. When developing CBR systems in this context, the challenge is to transfer knowledge from the case base to the vocabulary, similarity measures, and adaptation knowledge container.

Within this paper we will explain how CBR systems can be developed using existing data and creating an added value out of it. We explain how applying CBR as methodology can be integrated in existing processes and in the end will provide decision support capabilities and therewith reduce the effort of finding problem's solutions.

This paper is structured as follows: First we will describe the application domain of service reports for vehicles and discuss how existing data containing experiences can be used to create new applications. Section 3 explains first the underlying methodology followed by the processes executed in order to create the application. The following section 4 presents the evaluation based on first experiments followed by a related work discussion (section 5). The last section gives a short summary and an outlook on further activities in this area.

2 Service Reports of Vehicle Problems

Most vehicle companies provide service after delivering their machines to customers [10]. During the warranty period they are able to collect data about how and when problems occurred. They take this data for improving vehicles in different ways: collected data can go back in the product development process, it can be used for improving diagnostic systems to repair them at dealerships or in the factory and also educating service technicians repairing vehicles abroad. This is extremely important if vehicles cannot easily be taken back to factory, e.g. services for aircrafts or trucks.

Such machine manufacturers collect information about machine problems that are submitted by technicians containing machine data, observations and sometimes further correspondence between an engineer or analyst with the technician at the machine. In the end, these discussions usually come up with a solution - however, the solution is normally neither highlighted nor formalized and the topics and details discussed highly depend on the technician and what the Customer

Support asks for. That is the reason why cases that are stored for collecting Customer Support information can not directly be used for CBR. Therefore we will differentiate between Customer Support Cases (CS Cases) and CBR Cases. CS Cases contain original information collected during a discussion between Customer Support and the field technician, while a CBR Case contains only relevant information to execute a similarity based search on the cases. The CBR cases content represents each CS Case, but contains machine understandable information.

For the creation of CBR Cases, there were the following sources available:

- CS Cases created when a problem on a particular machine occurred
- Expertise that is in the field technician's and analyst's head, which they provide during a problem solving discussion.
- CS Solutions are textual descriptions of workflows created by analysts based on frequently occurring problems.

The CS Solutions are not solutions in terms of CBR, because they are not directly connected to a problem. The CS Solutions are FAQ-like descriptions how to solve a technical problem on a machine and are often created for an automatic diagnostic system. They provide a walk-through guidance for a technician. The quality of the CS Solution is very high because analysts looked through each CS Solutions and prepared diagnostic procedures. Reliable information is given and the diagnostic procedures are well structured. However, the CS Solution deal with one issue at a time and are only available for a certain number of problems. To increase the number of CS Solutions in order to have a higher coverage for occurring problems a high effort to create solutions is necessary. Another point why we decided to not use CS Solutions is the long time until the CS Solution is available for the technicians as well as the facts that solutions are not verified in the field and technicians might get stuck during a diagnostic procedure, because they contain many steps and not all events that might occur during a diagnostic procedure are covered.

CS Cases describe problems of a machine that cannot be solved by the technician. The cases describe a realistic situation in the field and contain information the field technician has access to while trying to solve the problem in the field. Further, the cases contain information about what was done in the end and what solved the problem. Those facts indicate that these cases should be reused, because the same problems might occur at the same type of machines and the work of analysts for solving the problem can be applied many times. Further, if the analyst would save time providing a solution that has already been applied, he can use his effort creating new solutions covering new problems. The disadvantages using CS cases are the facts that the cases describe a problem and their solution is hidden in unstructured text, the case content differs depending on the technician and cases that solve a problem right away with the first visit in the field are not available at all. Eventually, we decided to work with such CS cases, because they deal with experiential knowledge which perfectly matches the nature of CBR systems.

3 CBR Approach on Service Reports

3.1 Methodology

For making CBR system development as efficient and effective as possible the DISER methodology for developing CBR systems introduced by Tautz, Althoff and Nick [17,18,11] has been applied. It consists of four main development phases:

1. The creation of a *vision* that completely describes how to develop a CBR system on a highly abstract level. The main aspects in this phase are the identified *topics* and *goals* of the system. This phase also addresses whether *existing knowledge and infrastructure* can be accessed and reused. Further, the usage, creation and population of knowledge is defined in order to start a goal-oriented development process.
2. *Version 1* revisits the aspects of the previous phase and results in more detailed definitions that leads into a model and implementation of a first running system.
3. The *pilot* phase starts immediately after finishing the Version 1 phase. All features have been developed and can now be tested by users and according to the given feedback, they are refined by the developers. In this phase, change and maintenance management is established.
4. The *operation* phase starts with the system going online and providing its service to users. In this phase change and maintenance management starts.

For each CBR system to be developed the respective goals, based on the existing knowledge and infrastructure, have to be defined. Starting from these goals the subject areas have to be identified with which these goals can be achieved. For each subject area the scenarios for using the CBR system as well as the recording scenarios for filling the knowledge containers have to be defined. Based on this information the general and the case-specific knowledge is formally modeled, implemented, tested, and maintained.

Further, the research on mining textual, community based data has been carried out as a part of the further development of SEASALT, a domain independent architecture for creating knowledge-based systems based on experiences provided in web communities. SEASALT follows the idea of collaborating expert agents that join their expertise to solve complex problems. SEASALT describes on the one hand, how complex tasks can be modularized by creating so called topic agents and a knowledge line (a topic agent is a software agent that provides information on only one subtask) and also how to represent and extract knowledge out of an expert community and provide it for the overall knowledge-based system [13]. The work described in this paper focuses on the second part, the knowledge provision. The collection of machine data enhanced with the discussion between an analyst and a technician can be described as a community of experts that deals as knowledge source. In the following sections, we describe processing of the raw data in order to build a CBR system.

3.2 Source Data Analysis

After analyzing the available data with customer support and machine experts, we have created a case representation for the CBR system. Table 1 contains the structure of each case.

We have organized the attributes in various classes, however, each case in a complete instance of the given case representation. Basically each case can be divided in three parts: general case information, machine characterization and problem characterization.

Table 1. Service Report Case Representation

Concept: Meta			
Class	Symbolic	17 values	
Date Created		Date	
Status	Symbolic	3 values	
Season	Symbolic	7 values	
Origin	Symbolic	3 values	
Solution	String		
Concept: Problem			
Software	Symbolic	6 values	
Control Unit 1	Symbolic	68 values	
Control Unit 2	Symbolic	98 values	
Control Unit 3	Symbolic	63 values	
Control Unit 4	Symbolic	75 values	
Functional Code	Integer	min: 0, max: 10000	
Functional Area	String		
Brand Names	Symbolic	8 values	
Mechanisms	Symbolic	16 values	
Vehicle Parts	Symbolic	39 values	
Problem Description		String	
Concept: Location			
City	String		
State	String		
Topography	min: 0, max: 2000		
Concept: Vehicle			
ID	String		
Year	Integer	min: 1950, max: 2050	
Manufacturer	String		
Model	Symbolic	7 values	
Series	String		
Concept: Usage			
	Acres	Integer	min: 0, max: 20000
	Hours	Integer	min: 0, max: 5000
Concept: Discussion			
Key Part No	String		
Affected Control Unit	String		
Symptom	String		
Concept: Operating Conditions			
	Applicable	Boolean	
	Field	Symbolic	7 values
	Terrain	Symbolic	9 values
	Machine State	Symbolic	6 values
	Machine Loc	Symbolic	3 values
	Weather	Symbolic	3 values
	Activity	Symbolic	7 values
	Op Cond	String	

General case information (class Meta) contains general information about the case like the case *Class*, the *Date Created*, the case's *Origin* and whether one or more CS *Solution* is available or not. The Class is set by Customer Support analysts when the case is viewed the first time and describes the quality of the given information. The origin contains which way was used to submit the case: either using a software tool or calling the customer support by phone. The few cases submitted via phone were created during a phone call between the field technician and the Customer Support where the content of the call has been submitted later by the analyst. The *Season* is computed based on the date a case has been initially submitted. In the normal use case cases are submitted right after a problem has occurred and because of this fact we compute the season based on this date. The *Location* of a machine is described by the *City* and *State* where the machine is located as well as the *Topography*, which is the elevation of the City. This information is used to enhance the collected data with environmental information.

The second part describes the affected machine's characterization focusing on two aspects: On the one hand what kind of machine we are dealing with based on manufacturing information and on the other hand the current state of usage. The *Machine ID* is given for each machine and is usually a part of the case descriptions. Further, we use the Machine ID to extract the *model*, the *series*, the *manufacturing* factory as well as the *year* of production. Another characteristic feature of a machine is the usage until the problem appeared. The usage can be captured either as hours or acres. However, hours is the most common usage information among the CS cases.

The third part describes the problem description of the affected machine. Within this table the case-specific information is represented. The machine's problem description representation is divided in two parts: the problem description itself and the so called case text. The problem description contains a short description of the vehicle's fault. This information is reviewed by Customer Support and therewith the system can rely on that the given information describes the problem a particular case deals with. Since the problem description is a free text field it can contain all kinds of information, any of the following attributes can be populated: *Brand Names* contains names that can be used to describe a part of the machine. Further, we have different types of trouble codes which are modeled according the control unit they belong to. Like brand name, our system also scans the problem description for mechanisms, software names, and vehicle parts. We have modeled those information in different attributes rather than in one attribute to be able to assign more domain-specific similarity measures. The two attributes functional area and functional code are classifying the type of problem. This classification is assessed by Customer Support analysts and the CBR system uses this information to narrow down the potential matching cases to the relevant ones.

The Case Description attributes contain detailed information on how a machine's problem can be described. It contains observation information as well as trouble codes and information gathered during the machine's problem solving

process done by the technician. The population of this attribute varies depending on who is entering data as well as how data is entered. The *Symptoms* usually pick up on the problem description and describe briefly the detected failure. The *Operating Conditions* contain observations of the environment of the machine (like weather conditions or machine status). The *Affected Control Units* contain controller codes along with more observations. *Key Part Numbers* attributes contain information of the affected or to be replaced part of the machine. The attribute Discussion Text contains the whole discussion between Customer Support and the technician.

The starting point of this development is the analysis of the data available (historic experience items), data that is possible to be collected (potential experience items) and data that should be tracked/collected in the future (future experience items).

3.3 Vocabulary

The vocabulary is one of the knowledge containers and contains the definition of information entities and structures used to express knowledge, specifies the language within an application and contains a collection of terms that cover the relevant terms within an application domain.

The vocabulary of a CBR system covers the terms a CBR system can deal with. When we are referring to the vocabulary, we look at those terms that are modeled using symbolic value ranges and are based on a list of terms. Because of this fact, we will only cover those attributes that are represented as symbols.

The selection of terms that belong to an attribute are based on the occurrence of terms in the raw data. The goal during the modeling process is to cover the subject of an attribute with as much terms as necessary. However, the development of a CBR system is an incremental process and every time when new cases are inserted it should be ensured that the existing vocabulary is able to represent the newly inserted information. If the prerequisite is not fulfilled, the vocabulary has to be extended.

Depending on the attribute, the vocabulary can be created either by using common knowledge, value ranges that are given according the application domain, or based on specific domain dependent terms and abbreviations used within a company to describe products or processes. An attribute created based on common knowledge is for example Season: there are six different seasons (spring, summer, fall, winter, rainy and sunny season) available and we have created a list containing those terms. Value ranges or a given set of symbolic values are for example the usage of a machine or the series. Companies do have specifications for those kinds of attributes and they can be directly included in the vocabulary. Both of those types have in common that the value ranges are complete from the very beginning of the project and will only change after a major changes within the company or production (i.e. introducing a new machine series). However, obtaining the domain dependent vocabulary is more challenging, because there is no specific controlled vocabulary used. As described in Bach [2] it is essential, that the vocabulary covers as much relevant terms as possible.

Therefore we have used the following approach to create symbolic attributes that contain terms describing the information provided in the cases. Since we are dealing with unstructured text and each field technician can use the terms he would like to use, we choose a different way to first eliminate the terms to be modeled and second to create relations between two or more terms.

For the identification of the relevant terms, we executed this procedure on the raw data:

1. **Extraction of occurring terms:** We first executed a string tokenization algorithm so each term can be reviewed individually. Afterwards we removed stop words in order to only review descriptive terms. On these terms we applied a basic stemming algorithm. The result is a sorted list of terms that can be further processed.
2. **Classification of occurring terms:** We review the automatically created list of terms and created classes of terms.
3. **Discussing terms with experts:** Since we deal with a specialized vocabulary we then discussed the proposed classes and the contained terms regarding completeness, correct classification, known synonyms for the terms and relations between terms (for obtaining the initial similarity measures). The results were attribute values organized in tables and taxonomies with assigned similarity measures.
4. **Create the Knowledge Model:** Based on the classification attributes that contain the according terms were created.
5. **Refine the Knowledge Model:** We further reviewed the created models with experts and discussed examples ensuring that all relevant parts are covered.

At this point the vocabulary also contains some similarity information retrieved from the discussion with the domain experts.

3.4 Similarity Measures

Similarity measures are highly domain dependent and describe how cases are related to each other. The assessment of the similarity between two cases is calculated by an amalgamation function. We differentiate two types of similarity measures: Local Similarity Measures and Global Similarity Measures. Local Similarity Measures describe the relation between two symbols while the Global Similarity Measure represents the relation among attributes. The amalgamation function uses both similarity measures to compute the similarity between two cases. The next sections will discuss first the applied Local Similarity Measures, followed by the Global Similarity Measure.

Local Similarity Measures. Depending on the given raw data and available information, we picked a similarity measure. For attributes of the data type float and integer we used distance functions, for strings, we used string match or trigram matching and for symbolic value ranges we have used either similarity tables or taxonomies as they are described in [3].

Value ranges for numeric attributes are usually limited by the lowest and highest occurring value and initially we decided on a strictly monotonicly decreasing function. However, during discussions we have occasionally adjusted these measures.

Assigning similarities for symbolic attributes is more challenging, since you have to discuss the relation between the symbolic values. We usually started out creating a taxonomy and organizing terms within it incrementally. At the point when there are relations between different nodes, we moved on and created a similarity table to represent some kind of more dimensional attributes.

Global Similarity Measure. The Global Similarity Measure defines the relations between attributes. The weight of each attribute indicates its relevance within the case. We decided to use the weighted euclidean distance for the calculation of the global similarity. The development of each attribute's weight has been carried out by discussing the importance of each attribute when trying to find a similar case for solving a certain problem.

We decided to use a weight range between 1 and 5. The most important values are weighted with 5.0 and 4.0 depending on the experts opinion on which value they would focus on. The operating condition has also been named as an important attribute, but since there are seven different attributes which can be set while also a combination of up to seven attributes is possible (e.g. $activity = driving$, $applicable = true$, $terrain = hilly$ and $weather = dry$), we decided to decrease the individual weight. Usually one to four operating conditions attributes are set.

Further, one might point out that the symptom of a problem is weighted quite low, however, there is no clear distinction between the use of the symptoms text field and the problem text field. For the population of the attributes Brand Names, Control Units, Mechanisms, Software and Vehicle Parts the algorithm looks up for matching terms in the problem text field, the symptom text field as well as in the discussion text field.

3.5 Case Population

The CBR methodology supports us representing the available experience. In terms of CBR, each case can be seen as an experience item. A CBR case contains specific knowledge applied in a specific context in order to achieve a given goal. Transferring this idea to the project, each CS Case is subject to be transferred in a CBR Case since every case contains experience how to solve a problem for a certain machine. During the discussions with the Customer Support analysts we found out how knowledge contained in cases can be generalized and transferred from one case to another. This knowledge has been modeled in the vocabulary and similarity measure container.

Once the cases are created, they are organized within the case base. Each case can be described as a specific set of experiential knowledge that contains a problem and a solution description. The content of a case is based on the information that can be provided by a technician. Since we are only inserting

case information the CBR system can deal with, we match the information given with the available knowledge models, so the resulting cases are a subset of CS cases. The creation of the CBR cases can be seen as a mapping of raw data to the target cases. This mapping has to be done once the case base is created as well as for each query before the query can be submitted to the CBR system. We reuse the information extraction and mapping each time a query case is created. The mapping makes use of the vocabulary to extract relevant information and transforms the source data into CBR Cases. In this particular project we had to deal with different levels of data quality, because in some cases we had to deal with arbitrary text and other attributes contain trouble codes or company-intern terms that are very easy to extract. We have developed four different ways of how data is transferred from source to target data:

Copy Content. In this case we directly used the given data within the CS Case. There are only two different data types used: either string or date. Using this form of transformation, the case model has not been used.

Computation. In this case, the source data is used to compute an attribute value that is represented in the case model. We use this approach when we generalize attribute values in order to make them easier comparable. This has been applied to create two attributes: Based on the date a case was created we computed the season. The second attribute that we created based on the source attributes city and state is topography. Based on the city and state we used the GeoNames Web Service[1]. For the computation we used the city, state, country and the language the previous information is given in and retrieved a whole set of information about this place. From this information we extracted the elevation and added this to our case.

Model-Based IE. The Model-Based Information Extraction makes use of the vocabulary modeled previously. We use those terms and look them up in the according text field and store them in the case representation. An example for this type of extraction is the following problem description of a case:

> **gear box** does not respond - **rb7** software download

The case model contains the terms *gear box* in the attribute vehicle parts and *rb7* in software. The CBR system recognizes both terms, extracts those and assigns them to the according attribute ($software = rb7$, $vehicleparts = gearbox$).

New Composition and Model-Based IE. The most comprehensive extraction type uses the source attributes and computes/extracts a new set of information and then provides this result to do further *copying* or *model-based information extraction*. This has been done with the Case Description

[1] `http://www.geonames.org/export/ws-overview.html`, GeoNames is a geographical database accessible via web services that contains over 2.8 millions populated places.

attribute, because there are more structured information included in one large text attribute. We first re-created the structure and then applied the information extraction.

The case problem description in general contains information that can be initially provided by the technician and the solution is suggested in the discussion between the Customer Support analyst with the technician. The case representation does not contain each source attribute, however, when the cases are presented to the user, they are included to provide the most comprehensive information.

3.6 Rapid Prototyping Tool myCBR

The CBR tool myCBR has been mainly developed to provide an easy-to-use interface for rapid-prototyping of CBR applications [16]. Its main focus is providing a easily manageable tool for creating CBR prototypes, especially similarity measures. Further, myCBR supports the import of cases stored in csv files and testing the created vocabulary and similarity measures.

We have used the myCBR interface to develop the knowledge model and talk with the domain experts about the definitions and relations of value ranges. myCBR provides the most common similarity measures and as an open source tool additional ones can be included as well. After creating the vocabulary and similarity measures we imported the cases from a csv file. In advance, we created cases as described in section 3.5 and loaded them into myCBR. Afterwards first retrieval tests are possible and we started first refinements on the similarity measures. The included retrieval mask and the handling of multiple case bases allows the knowledge engineer to carry out various tests.

For demonstrating the capabilities of the CBR system, we used the myCBR SDK to access the CBR system and built a JSP interface. Therewith, we were able to install the application on a web server in the company and possible users can easily access the application via a web browser. For the presentation of the results we decided to use the source data the users are familiar with and extended this with a case representation that comprises the CBR cases.

4 Experimental Evaluation

The cases and processes we deal with are rather complex so we decided to do a qualitative evaluation using experts to measure the effort of developing such a kind of CBR system. Today the analysts are not able to search within all attributes. They are only able to search within certain attributes. The problem that we have used for the detailed result analysis, for example, could not be solved with the currently available sources. In this case, the analyst or an engineer has to create a solution for this problem.

For the first test, we included 953 cases and each case can be represented using up to 40 attributes depending on the detailedness of each case. About the half of the attributes are represented as symbols and have an individual local

similarity measure. We have evaluated three different aspects of our approach that came up during carrying out the project.

4.1 Extending the Case Base

The initial development has been based on a set of 953 cases covering one certain class of cases. Those cases dealt with a particular area of a machine. Extending the case base does also require an extension of the vocabulary (of symbolic attributes), value ranges (for numeric attributes), and possibly the similarity measures. After a first evaluation of the created case base, the customer requested an extension of the case base using 145 new cases that are slightly different in the problem type and extended the scope of the application. This requested extension was carried out very quickly and without any problems.

First of all, we had to analyze the data as described in section 3.2, before the extension of the vocabulary could be conducted. The term extraction is completely automated and all we had do to was inserting about 20 new terms in our taxonomies. Afterwards the csv file using the knowledge models is created and loaded into the system.

4.2 Comparing with Utility

For evaluating the quality of the CBR system, we had a customer support expert who created an example question that describes a problem, which occurred in the past, but its solution is not covered in manuals, nor the exact same case is available in the training case base.

The task for the CBR system was to come up with the best solution. The comparison of the results can only be measured by rating the quality and for this purpose the customer support experts developed their own measure to rate the utility of a case.

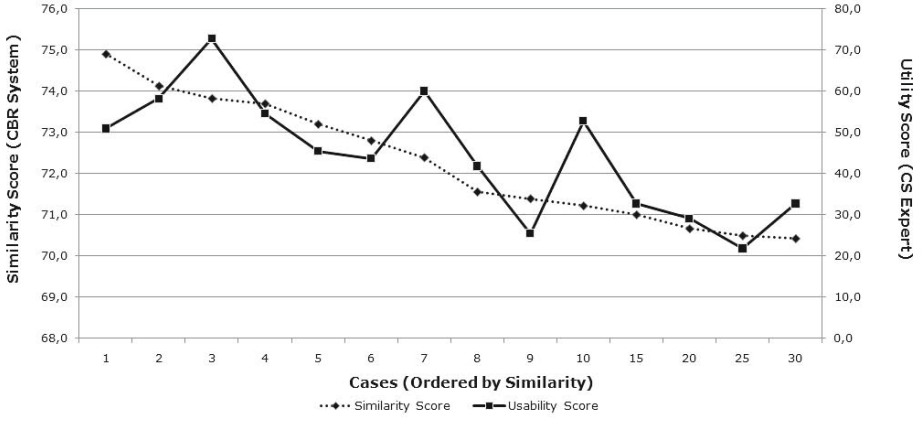

Fig. 1. CBR Similarity vs. Utility

They created 11 criteria describing the utility and rated them on a scale from 0 (this criterion is not covered in the case) to 5 (this criteria is correctly covered). The criteria used are the relevancy to the problem, how explicit the problem is described and a solution is presented. Further, the understandability of the cases and references to further readings are scored. The expert reviewed the top 10, the 15th, 20th, 25th and 30th case and rated them.

Figure 1 shows the results of the evaluation. The y-axis contains the relevance scores. On the left part you can see the similarity score of the CBR system and on the right part you have the utility score based on the rating of the expert. The x-axis contains the case rank according to the CBR system. The tendency of both, the CBR system and the expert, is alike, however the expert would have come up with a different order. Further, within the top 10 results of the CBR system were the most relevant cases to the given query. This shows that the CBR system containing semi-automatically extracted knowledge, which was slightly refined during discussions with experts produces results that match an expert's opinion. Even if the best match did not show up in first place, our tests showed that the first ten results always contained the best matching case.

4.3 Effort of Including Experts

The approach presented in this paper is aiming at as many automation during the build-up of the CBR system as possible. The process included discussion with experts, but did not require that experts had to create the knowledge models themselves. We carried out six discussions as telephone conferences or tele-presence meeting along with two two-day workshops. During the discussion there were usually two experts present. This shows that the approach makes use of the experience provided in the source cases and technical manuals from the very beginning throughout the entire project.

5 Related Work

Case-based diagnosis is a traditional topic in CBR research. The task of our approach can be compared to the diagnosis tasks described by Lenz et. al. in [7,8]. However, we decided to use a structural CBR approach, because the overall goal was to create diagnosis automatically by further developing the acquired knowledge. A Textual CBR approach could not be extended this way so easily.

The number of features and the associated values of our system are comparable to Koton's CASEY [6], a medical diagnosis system. However, our approach uses existing service reports and builds a CBR system on them while all knowledge in CASEY was created to be applied in the CBR system. Besides supporting medical diagnosis, help-desk support systems like HOMER [4] or CASSIOPÉE [5] can also be related to our work. Both approaches are Structural CBR system.

The raw data which has been created as a collaboration between technicians and analysts can be described as community experiences captured during expert discussions. Also, the type of experience we deal with according to Plaza and

Baccigalupo [12] is how-to experience, however we did not focus on the process aspects like Minor et. al. [9] do, we focused more on the extraction and provision of the whole items. Given the fact of the fast and broad dissemination of web communities and therewith the availability of huge amounts of experience knowledge, it is very promising integrating experiences in CBR systems, because the underlying methodology of CBR relies on previously made experiences. It is important to use, combine and further develop technologies that have already been applied on the Web (2.0) together with standard technologies. Therefore software engineers for CBR systems should create web-based systems, because that might be the only way to receive feedback [15]. Our approach uses web technologies to disseminate the application within a company and we have tried to be as less restrictive as possible when searching for a case. For example, the most criteria can be described in free text and the result list contains the original data. This ensures that the whole content of a case is provided to the user, if necessary.

6 Summary

This paper presented the outcomes of a research project that applies CBR as a methodology to make use of existing data. We first introduced the application domain and described the characteristics of service reports for machines before we explained how we transferred existing data to cases and filled the knowledge containers during the development of the CBR system. The evaluation of our approach shows that it can definitely be applied in this scenario. Further, during the development process, the experts confirmed that they learned new aspects of their own data and how it can be looked at in different ways depending on the goal of a system. The CBR system we have developed also demonstrated the capabilities of the technology and at the same time identified new potentials. For example, collecting more structured data can lead to more automation towards an automatic machine diagnosis and higher confidence in a proposed solution. Another aspect is collecting feedback and start to learn or improve similarity measures, which will also lead into more precise retrieval results. Of course, the more knowledge we put into the similarity measures the higher the retrieval precision will be.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications 1(7) (March 1994)
2. Bach, K., Hanft, A.: Domain modeling in tcbr systems: How to understand a new application domain. In: Wilson, D.C., Khemani, D. (eds.) ICCBR 2007 Workshop Proceedings, WS on Knowledge Discovery and Similarity, pp. 95–103 (2007)
3. Bergmann, R.: Experience Management: Foundations, Development Methodology, and Internet-Based Applications. LNCS (LNAI), vol. 2432, p. 25. Springer, Heidelberg (2002)

4. Göker, M.H., Roth-Berghofer, T.: The development and utilization of the case-based help-desk support system homer. Engineering Applications of Artificial Intelligence 12(6), 665–680 (1999)
5. Heider, R.: Troubleshooting cfm 56-3 engines for the boeing 737 using cbr and data-mining. In: Smith, I., Faltings, B. (eds.) EWCBR 1996. LNCS, vol. 1168, pp. 512–518. Springer, Heidelberg (1996)
6. Koton, P.: Reasoning about evidence in causal explanations. In: 7th National Conference on Artificial Intelligence, pp. 256–263. AAAI Press, Menlo Park (1988)
7. Lenz, M.: Defining knowledge layers for textual case-based reasoning. In: Smyth, B., Cunningham, P. (eds.) EWCBR 1998. LNCS (LNAI), vol. 1488, pp. 298–309. Springer, Heidelberg (1998)
8. Lenz, M., Busch, K.H., Hübner, A., Wess, S.: The simatic knowledge manager. In: Aha, D., Becerra-Fernandez, I., Maurer, F., Muñoz-Avila, H. (eds.) AAAI 1999 KM/CBR Workshop, pp. 40–45. AAAI Press, Menlo Park (1999)
9. Minor, M., Bergmann, R., Görg, S., Walter, K.: Towards case-based adaptation of workflows. In: Montani, S., Bichindaritz, I. (eds.) ICCBR 2010. LNCS, vol. 6176, pp. 421–435. Springer, Heidelberg (2010)
10. Müller, T., Lange, K., Breuer, A., Krieger, O., Form, T.: Automatic and expierence-based diagnostics using distributed data and neural networks. In: 12. Internationaler Kongress Elektronik im Kraftfahrzeug, pp. 593–605. VDI Verlag (2008)
11. Nick, M.: Experience Maintenance through Closed-Loop Feedback. Ph.D. thesis, TU Kaiserslautern (October 2005)
12. Plaza, E., Baccigalupo, C.: Principle and praxis in the experience web: A case study in social music. In: Delany, S.J. (ed.) ICCBR 2009 Workshop Proc. Workshop Reasoning from Experiences on the Web, pp. 55–63 (July 2009)
13. Reichle, M., Bach, K., Althoff, K.D.: The SEASALT Architecture and its Realization within the docquery project. In: Mertsching, B., Hund, M., Aziz, Z. (eds.) KI 2009. LNCS, vol. 5803, pp. 556–563. Springer, Heidelberg (2009)
14. Richter, M.M.: Introduction. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.D., Wess, S. (eds.) Case-Based Reasoning Technology. LNCS (LNAI), vol. 1400, pp. 1–16. Springer, Heidelberg (1998)
15. Smyth, B., Champin, P.A., Briggs, P., Coyle, M.: The case-based experience web. In: Delany, S.J. (ed.) ICCBR 2009 Workshop Proc. Workshop Reasoning from Experiences on the Web, pp. 74–82 (July 2009)
16. Stahl, A., Roth-Berghofer, T.R.: Rapid prototyping of CBR applications with the open source tool myCBR. In: Althoff, K.-D., Bergmann, R., Minor, M., Hanft, A. (eds.) ECCBR 2008. LNCS (LNAI), vol. 5239, pp. 615–629. Springer, Heidelberg (2008)
17. Tautz, C., Althoff, K.D.: A case study on engineering ontologies and related processes for sharing software engineering experience. In: Proc. of the 12th Internal Conference on Software and Knowledge Engineering, Chicago, USA (July 2000)
18. Tautz, C., Althoff, K.D., Nick, M.: A case-based reasoning approach for managing qualitative experience. In: Proc. 17th National Conference on Artificial Intelligence (AAAI 2000), Workshop on Intelligent Lessons Learned Systems, pp. 74–81 (2000)