Developing Case-Based Reasoning Applications Using myCBR 3

Kerstin Bach[1,2] and Klaus-Dieter Althoff[1,2]

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

Abstract. This paper presents the Open Source tool myCBR which has been re-implemented as standalone application with a designated application programming interface that can be used as plug-in for various applications. We will introduce how knowledge according to Richter's knowledge containers can be modeled and how myCBR has been successfully applied within various applications. Especially we introduce novel features of myCBR that support knowledge engineers developing more comprehensive applications making use of existing knowledge such as Linked Data or User Generated Content. The applications presented in this paper present the high variety how CBR can be applied for web-based and mobile technologies as well as configuration, diagnostic or decision support tasks.

Keywords: Case-Based Reasoning Tools, Knowledge Container Development, Case-Based Reasoning Applications, Open Source Software.

1 Introduction

Researching for novel Case-Based Reasoning approaches requires a test bed where users can easily create CBR applications using on the one hand an intuitive user interface and on the other hand including CBR as modules in more complex software systems. The re-implementation of myCBR[1] in Java is offering both to its user. For the re-implementation we aimed at keeping the user friendly interface, while directly accessing the XML-based knowledge representation.

The underlying idea of Case-Based Reasoning is reusing previous cases for solving future problems [1]. Following this principle we should also capture our experience when creating CBR systems and provide it for future use. This is one

[1] Throughout this paper, when referring to myCBR we are always speaking of myCBR 3 (http://mycbr-project.net/preview)

B. Díaz Agudo and I. Watson (Eds.): ICCBR 2012, LNCS 7466, pp. 17–31, 2012.

reason for developing the Open Source tool myCBR, which is available under the GNU General Public License. At the German Research Center for Artificial Intelligence we are aiming at using myCBR within projects and therewith further develop features that are given back to the research community.

In this paper we will on the one hand show how knowledge required to develop a CBR system can be built using the user interface, how applications have successfully been developed based on the Software Development Kit (SDK) and also how the existing myCBR implementation has been extended. The myCBR functionalities are explained along the knowledge containers introduced by Richter [22]. They differentiate between compiled (vocabulary, rules and similarity measures) and interpreted (cases) knowledge. For each knowledge container we describe the features that are supported by the core myCBR tool and possible extensions that might be useful for certain applications.

myCBR as a tool for creating CBR systems should be able to cover a high variety of tasks such as decision support, diagnosis, planning, etc. Especially when building knowledge-based applications two aspects have to be considered: during the definition phase the knowledge bases have to be created and discussed with experts. Therefore a user interface is crucial for making this knowledge transparent. On the other hand for running the CBR system we need an engine that makes use of previously created knowledge. myCBR offers both – the first one is referred as myCBR GUI and the latter one as myCBR SDK. The GUI makes use of the SDK and wraps a user interaction interface around it.

This paper is structured as follows: First, Section 2 will introduce how the four knowledge containers are addressed within myCBR and how novel features extend the tool for its applicability in a higher variation of application scenarios. The second part, Section 3, will showcase, based on seven myCBR applications, how the tool can serve in an industrial and scientific context. In the following related work section we will have a closer look at other, freely available, CBR tools such as FreeCBR, jCOLIBRI or eXiT*CBR. The last section gives a short summary and an outlook on further activities in this area.

2 Knowledge Container Development

Figure 1 shows the general architecture of myCBR. The left hand side describes the core components including the data import, the knowledge containers, the explanation component as well as the retrieval module. While the model component holds the vocabulary in various types of attribute descriptions, the similarity measures are described independently and connected to the model itself. The case base provides the case organization as well as case addition and deletion. The transformation component is only available in myCBR's OSGi release, but offers adaptation and completion capabilities as they are described in Section 2.2.

Additionally we developed extensions for making myCBR applicable to a broader range of applications. These extensions currently focus on Knowledge Engineering and Knowledge Acquisition tasks. The *myCBR GUI* is the user interface for creating myCBR applications by filling the knowledge containers.

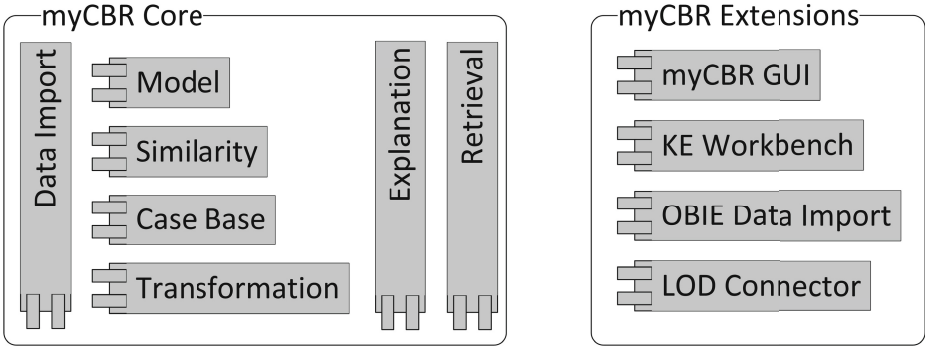


Fig. 1. myCBR Architecture

The other three extensions focus more on extracting relevant information from unstructured data and integrating them (semi-)automatically in the according knowledge containers.

2.1 Vocabulary

The vocabulary can be either built from scratch or by making use of existing data. The CBR applications we are targeting at with myCBR are structural CBR applications with a flat or object-oriented case representation (i.e. as described in [4]). Depending on the application and available data, Linked Data seems a good starting point building a proper, well-covering vocabulary. The myCBR SDK (myCBR Core in Figure 1) has an extension, called *LOD Connector*, which can access Linked Open Data sources such as DBPedia[2] or Freebase[3] for building taxonomies. When accessing linked data for building a vocabulary we target at concepts and their relations for eventually representing cases. While the concepts will serve as attribute values, the relations will fill the similarity measure container (see Section 2.3). For further use, the provenance of each value is stored in the concept explanations [24]. When building a taxonomy, the similarity between values is computed as follows: We start with a base similarity of $sim_0 = 0$ (by default), which is assigned to the root node. For each hierarchy level i, the children, grandchildren, etc., we compute the similarity to the root node as follows:

$$sim_i = 1 - \frac{1}{2^i} \tag{1}$$

The resulting taxonomy will then be organized comparable to the following skeleton, which contains for each parent node the similarity for all children in common [6]:

[2] `http://dbpedia.org`
[3] `http://www.freebase.com/`

```
+ root [sim_0 = 0]
  + Children_A [sim_1 = 0.5]
    - Grand_Children_AA
    - Grand_Children_AB
  + Children_B [sim_1 = 0.5]
    - Grand_Children_BA
    + Grand_Children_BB [sim_2 = 0.75]
      - Grand_Grand_Children_BBA
      - Grand_Grand_Children_BBB
  + Children_C [sim_1 = 0.5]
...
```

Another approach for creating the vocabulary is the fact that the myCBR user interface supports knowledge engineers defining the vocabulary. The myCBR graphical user interface (GUI) enables creating attributes with numeric and symbolic value ranges. From our experience, this user interface also supports the discussion with the domain experts, because they can see the knowledge model and provide insights [4].

2.2 Rules

So far, completion and adaptation rules are not deeply integrated in myCBR. However, we decided to create an integration of the industrial-strength rule engine Drools. In advance of this decision, we took a look at three different rule engines: JBoss Drools [7], jRete[4] and jRuleEngine[5]. We evaluated their suitability in the context of myCBR: on the one hand, according to the functionality they offer and, on the other hand, the integrability of the rule engine in the existing tool. Since myCBR is a freely available tool, only rule engines were examined, which are also freely available. Further, only Java-based rule engines are relevant allowing a straightforward integration into the Java based myCBR. All three use the Rete algorithm for optimization of the rule processing.

JBoss Drools Expert. Drools Expert is a rule engine of the JBoss community and part of the business logic platform. This platform consists of several modules and Drools Expert is the only relevant module in the myCBR rule engine context. The rules are defined in a proprietary rule language, the Drools Rule Language (DRL) and stored in a proprietary file format, which is based on DRL. Drools Expert supports many functions that are applicable for a Rule Engine in the area of CBR. Further development is necessary for handling symbolic attribute descriptions, which is required in a model based adaptation. Also other functionalities like set operations or similarity assignments have to be implemented additionally. Arbitrary objects can also be loaded into the Rule Engine and used by all rules as global objects. In addition, Drools Expert can be integrated into the myCBR GUI via OSGi and is offering most of the features

[4] `http://sourceforge.net/projects/jrete/`
[5] `http://jruleengine.sourceforge.net/`

required for completion and adaptation rules along with its open and modular framework.

None of the three rule engines can be recommended for the use without any limitations for myCBR. jRete and jRuleEngine by far do not offer the necessary functions and the customization would require an extremely high effort. Drools, however, offers a large part of the necessary functionality and can be connected with myCBR via the OSGi interface. Integrating Drools in myCBR meant having two types of rules to be implemented. On the one hand we have completion rules that run on the complete case base as a sub-process of loading the case base. They are built upon domain knowledge and enable the CBR system deducing attribute values from explicitly given features in a case or a query. Within this sub-process additional information is loaded, for example if you take cooking recipes as a case, completion rules can add the type of meal based on the ingredients and preparation methods [18]. Furthermore, each user query is also enriched with information using completion rules. The other type of rules, adaptation rules, are only applied to a subset of cases for performance reasons. Adaptation rules can become very complex and that is the reason why we only use them on the top 5/10/20 cases in order to adapt them to the user's preferences. Before Drools rules can be applied to a myCBR case base we first had to transform the myCBR SDK in an OSGi structure providing the required services.

Completion Rules. The process of applying Drools rules as completion rules is pictured in Figure 2. After the initial myCBR case base has been loaded the completion rules are loaded from a CSV file. Completion rules are simple if-then-rules that are applied to each case, where the if clause describes the condition and the then part the action. If an attribute value is already set it replaces the existing or adds another value.

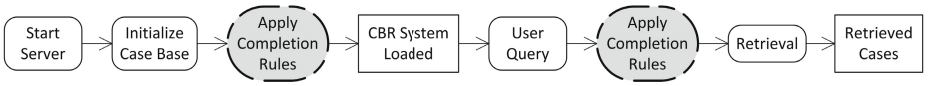


Fig. 2. Completion Rules for myCBR based on Drools

Adaptation Rules. Adaptation rules are more complex since the conditions as well as their according actions might cause time-consuming changes on a case, because depending on the underlying knowledge models, for instance similarity tables, taxonomies, calculations, adequate substitution candidates or value changes have to be retrieved along with the conditions that specify the circumstances when a rule can be applied. Afterwards the case itself gets adapted. The idea behind this way of adaptation is depicted in Figure 3.

One of the challenging aspects determining adequate adaptation candidates is making use of existing knowledge models. For instance when using taxonomies for similarity assessment we have sibling nodes and parent nodes. Depending on the knowledge engineering strategy, one has to define how substitution candidates are selected. Afterwards the rule also has to check whether the node is (a)

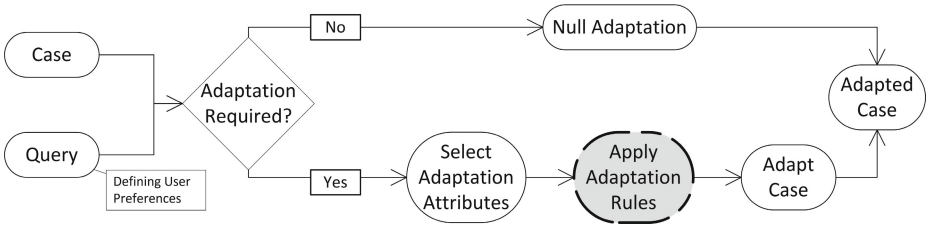


Fig. 3. Adaptation Rules for myCBR based on Drools

artificial and should not be used or (b) it fits the given user preferences. Artificial nodes are common in complex knowledge models. They represent connections between nodes that implicitly exist, but for which no instance is present (either caused by incompleteness or circumvented complexity/overfitting). Drools, does not support that feature in particular, however, it allows the access to knowledge models and the following steps describe how we currently compute substitution candidates (assuming that the values to be substituted are known):

1. Identify the taxonomy T in which the value n_{target} to be adapted is situated.
2. Identify the parent node n_{parent}.
3. Identify all children nodes $(n_{sibling_1}...n_{sibling_n})$.
4. Remove all artificial nodes of $n_{sibling_1}...n_{sibling_n}$ until only instantiable nodes exist.
5. Remove all n_x of $n_{sibling_1}...n_{sibling_n}$ that do not fit the user's preferences provided in the initial query.
6. Select the adaptation candidate with the highest similarity to n_{target}.

The identification of artificial nodes has been carried out by using myCBR's explanation capabilities, which allows the user to define so-called concept explanations for every entity in the model [24]. In this case we tagged artificial concepts in order to be accessible by the adaptation rule. Besides the adaptation based on knowledge within the knowledge containers, the rule engine based on Drools also allows more complex adaptation approaches that creates solution from case skeletons or even from scratch.

2.3 Similarity Measures

The similarity measures provided in myCBR do cover simple data types like Int, Float or Double for numeric attributes or String for textual attributes. myCBR also provides table similarities or taxonomies for symbolic value ranges. For numeric attributes myCBR additionally supplies also the similarity assessment based on the difference and quotient of values.

A very powerful similarity modeling approach provided by myCBR are similarity measures for set attributes. They allow more than one value per attribute

and therefore the adequate handling is required. An example of the successful application is discussed in Section 3.1. A more detailed view on how similarity measures can be modeled in myCBR is given in [25].

2.4 Cases

Cases are given by the application domain and have to be transformed or indexed in order to fit the given case representation. Further, especially for e-Commerce or decision support scenarios on a common domain they can also be acquired from web sources. Especially when the source is User Generated Content some type of Information Extraction (IE) should be applied. Currently myCBR Core does not support any of these capabilities, however, we have created the *OBIE Data Import* extension (see Figure 1) for using IE tools when building the case base. Up to today we can use either SCOOBIE [2] or GATE's ANNIE component [9]. Both approaches use ontology-based IE, because this allows making use of Named Entities (NE) [12]. Cunningham [10] lists the requirements for ontology-based IE that we will also apply: Instead of plain gazetteers, we will use ontologies for which there is the main challenge populating the ontology. However, since we assume that the vocabulary has been defined as a knowledge model, for example a taxonomy, we initialize the IE process using that model. The result is a populated case base.

IE with SCOOBIE. SCOOBIE is an IE tool developed at DFKI that uses symbolic descriptions, especially RDFS[6] for describing, learning and further developing domain ontologies. SCOOBIE can be queried using SPARQL[7] and this is how the myCBR IE component makes use of SCOOBIE: we transform the myCBR knowledge model in an RDFS graph and during the process of loading cases we carry out IE that matches terms and populates cases.

IE with GATE. The GATE extension on the other hand makes also use of ontologies, but also applies more IE technologies. GATE's IE tool ANNIE [11] uses the so-called OntoRoot Gazetteer to create gazetteers based on ontologies, which are then applied to tokenized and NE-tagged data sources. IE based on ANNIE is more flexible because of its plug-in structure and depending on the provided data, additional NLP technologies can be applied. Furthermore, we tested the performance of both approaches measuring the computing time based on two knowledge models.

To compare these two IE approaches we carried out some performance tests on a MacBook with an Intel Core 2 Duo (2 GHz) processor and 4 GB RAM. Further we had two different knowledge models, model 1 contains 52 attribute values, model 2 has 1,482 attribute values, and we used three case bases that had to be indexed using model 1 and model 2. All models and case bases contain cooking recipes. Case base 1 (CB-1) contains only pasta recipes and model 1 has been created for that particular case base. CB-1 also contains reviewed and

[6] `http://www.w3.org/TR/rdf-schema/`
[7] `http://www.w3.org/TR/rdf-sparql-query/`

Table 1. IE benchmark for two knowledge models (seconds for ontology based IE and case population)

Case Base	Model 1		Model 2	
	SCOOBIE	ANNIE	SCOOBIE	ANNIE
CB-1 (108 cases)	203.87	25.83	323.72	21.64
CB-2 (100 cases)	69.33	10.54	87.23	24.77
CB-3 (1000 cases)	139.70	29.19	247.71	40.90

well-structured raw data. Model 2 contains more comprehensive cooking recipe knowledge, which targets at any type of recipe. Case bases CB-2 and CB-3 are arbitrary recipe collections of different sizes.

As Table 1 shows, the results integrating ANNIE outperform SCOOBIE. The quality of the extraction did not substantially differ. Another obvious effect is that the more matches we found, the more effort time the case population takes. This explains why the Model 1 – ANNIE population of CB-1 is much slower (more than twice the time) than the population of CB-2. Using the larger model, which contains almost 30 times more values, produces quite similar run times. Having this fact in mind and looking at the run times for populating CB-3 it shows that the number of cases populated influences the run time of case population most.

3 CBR Applications Based on myCBR

After introducing myCBR features as well as add-ons for an easier knowledge container population, this section will focus on applications we have recently built using myCBR. Each of the following applications uses core myCBR features, which are the vocabulary, multiple similarity measures, one – often more – case bases and a sequential search in a non organized case base. Depending on the applications' requirements we extend these existing features, for instance for populating cases for a case base, including Linked Data for building the vocabulary or creating cases, pre-processing data using IE, adding rule engines, or applying constraints. myCBR core is provided as a jar archive that provides CBR capabilities. The required processes and application logic has been developed independently and usually based on the examples given on the myCBR website.

Table 2 gives an overview of the applications we will discuss. The applications are grouped by their task and each row shows the amount of attributes (symbolic, numeric or textual), the used similarity measures, what type of rules is used, how many cases are included and which type of front end we provide to access the systems.

The core role within these applications is played by structured CBR and they show how manifold myCBR applications can turn out. About the half of the applications was created by Bachelor's and Master's students under supervision of a myCBR team member, the other part was implemented in research projects.

Table 2. Overview of myCBR-based applications

Name	Vocabulary	Similarity Measures	Rules	Cases	Front-End
Decision Support					
CookIIS	21 attributes, 1482 symbols	taxonomy, exact match, n-grams	178 completion rules, model-based adaptation	1489	mobile, web
EatSmart	13 attributes, 1323 symbols	taxonomy, exact match	none	272	web
myCamera	15 attributes, 38 symbols	taxonomy, polynomial, distance, exact match	none	624	web
FinancialDS	21 attributes, 140 symbols	taxonomy, table, cyclic, polynomial, distance, exact match, n-grams	4 simple completion rules	70	web
Configuration					
PC-Config	45 attributes, 116 symbols	taxonomy, polynomial, distance, exact match	hard coded constraints	488	web
Diagnosis					
Service Cases	31 attributes, 445 symbols	taxonomy, table, cyclic, polynomial, distance, exact match, n-grams	none	924	web
Decision Support & Information Composition					
docQuery	65 attributes, 1081 symbols	taxonomy, table, cyclic, polynomial, distance, exact match, n-grams	none	1061	proprietary

3.1 myCBR Applications

CookIIS is a CBR cooking recipe engine that showcases the strength of CBR in an easily understandable domain, which affects everyone once in a while: what to cook with the ingredients I love – while respecting allergies or dietary practices. CookIIS is developed since 2007 [13] and has been reimplemented recently using myCBR [5]. Further, it is the first time we used Drools as it is described in Section 2.2. This novel implementation also closes the loop of 4R [1] since it covers new features that collect feedback (Revise) and include new cases semi-automatically (Retain). From the user interaction point of view, CookIIS now made its way closer to the oven since it has an Android interface. The

current version of the web interface as well as the mobile app can be found here: `http://www.dfki.de/~bach/cookiis.html`.

EatSmart is a CBR application that supports its users dealing with a *Metabolic Typing* conform nutrition. It picks up the idea of CookIIS, but is personalized and works with an individual training and eating plan. Usually only a menu for two or three weeks is given to *Metabolic Typing* customers. However, EatSmart increases the variety of recipes that fit in a certain nutrition plan and on the other hand points out if a user is looking for recipes or ingredients that are forbidden in the given nutrition plan. EatSmart is currently available in German only and can be found here: `http://cbrdemo.kl.dfki.de/EatSmart`. EatSmart showcases that CBR applications can make life easier by including intelligence into systems. The evaluation of this application with *Metabolic Typing* customers brought good results, because it was easier, especially for new users, to get familiar with ingredients and dishes that are allowed and those, which should be avoided.

myCamera differs from the before mentioned application since the user scenario – selecting an adequate digital camera – is less complex. However, myCamera showcases how different user interfaces can be implemented. On the one hand, in the simple search mode there are four questions to be answered and based on the given answers the underlying attributes are set. The answers are statements that describe how the camera will be used. On the other hand, the detailed search lets the user select eleven features that initialize a similarity-based retrieval. The novelty of myCamera are the cases that are retrieved from freebase. We query Linked Data and then populate the case base automatically. myCamera can be accessed via `http://cbrdemo.kl.dfki.de/myCamera/`.

PC-Config showcases how multiple case bases, in this particular application six: RAM, CPU, HDD, Graphic Card, Main board, Previous Configurations, can be integrated in order to configure valid PC systems. For the configuration either previous configurations can be recalled and revised to initialize a query or users can specify their preferences and the application starts configuring. The configuration is based on the selected main board, which sets the constraints for the subsequent components. Each case base is queried individually and the best matching cases are integrated. If the best match does not fit, PC-Config takes the second, third, etc. The application is in German only, it can be found here: `http://cbrdemo.kl.dfki.de/PCKonfig/configpage.jsp`. Besides addressing a configuration task, this application also makes use of Linked Data: the attribute values where obtained from Freebase and automatically populated in the myCBR knowledge model.

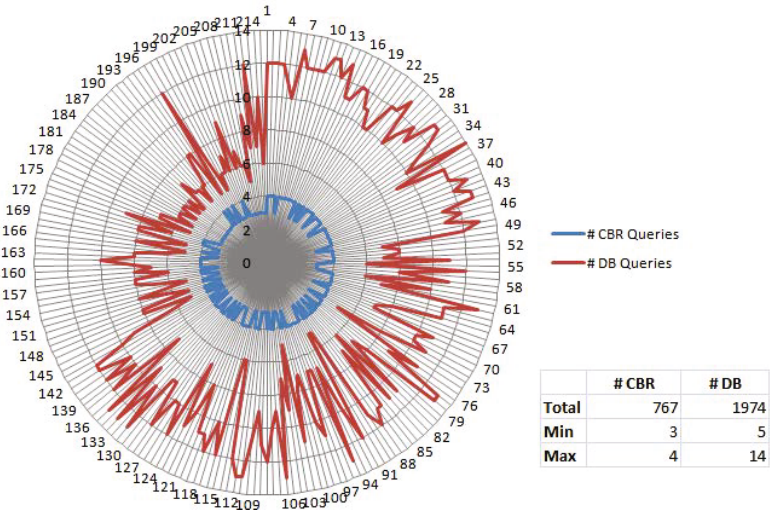
FinancialDS is applied in the domain of suggesting customers best fitting financing offers for certain goods. In this demonstrator application up-to-date financial plans and offerings have to be combined in order to create competitive offers for customers. This application partially makes use of Linked Data for populating the companies product names in the knowledge model and on the other hand basic, hard coded completion rules were used for Information

Extraction and attribute population while loading the case base. FinancialDS uses company internal guidelines for creating those cases that will then be presented to customers as product offers, so this approach uses general knowledge (guidelines) to produce an individual piece of information (product offer). Furthermore in the very beginning we only had a few guidelines available and only little expert knowledge on how to design similarity measures. The customization of guideline allowed us to create a proper case base and the high coverage of cases overcame the similarity measure shortcoming. According to Richter [22] this is one way how the knowledge container principle can adapt to various degrees of formalization.

Service Cases is a machine diagnosis application that was presented in detail in [4]. Summarized Service Cases make use of protocols between first and second level support of machines for providing faster, more efficient help in case a machine breaks down in the field. This application describes a classic scenario in which a well-organized collaborative knowledge base can be used in multiple areas – for maintaining warranty claims, improving the product by identifying weak points, and providing lessons learned by one participant to a broader audience. This type of diagnosis surely requires enough stakeholders, however, the scenario can be applied in many comparable domains. For instance, we have used a similar approach in a prototype for an intelligent office scenario where maintenance requests for ships were processed in a case-based way in order to save time from the invoice until an expert is sent out.

docQuery is a travel medicine application based on the SEASALT approach [21], which picks up the idea of collaborating multi-expert-systems [3]. The implementation does not have a web-based or mobile front end, which are currently developed. However, it integrates myCBR in JADE and currently only communicates via FIPA-ACL [8]. Within docQuery CBR is applied in two ways: First, as topic agents representing an expert in a particular area; second, as coordination agents that make use of previous query paths [17]. docQuery shows how myCBR can be applied in a distributed, agent-based architecture. The docQuery knowledge model is an object-oriented model, where each class has its own case base. When loading the topic agents, we initialize this set of case bases that is represented by the agent. The benefit of using CBR as underlying knowledge-based system is clearly the ranking of best matching results and the easy handling of multiple attributes. For example, when we are querying for diseases that occur in a certain region, we usually need information for one up to seven diseases. Querying a data base would mean sending seven requests – using myCBR's multiple value attribute means sending one request and receiving a list of diseases where the first seven hits have the same similarity and are relevant for further processing.

Figure 4 illustrates this effect for the queries required in docQuery to obtain information for each of the 214 countries (no 1-214 in the Figure) included in the system. In total CBR topic agents had been queried 767 times while the number of data base requests would have been 1,974. In this figure, the inner line represents the CBR queries that vary between 3 and 4 queries for each



	# CBR	# DB
Total	767	1974
Min	3	5
Max	4	14

Fig. 4. docQuery: CBR vs. DB queries for diseases

country – one query for each populated attribute. The outer line shows DB requests based on the same attributes. They require between 5 and 14 queries.

4 Related Work

Freely available CBR tools are for instance FreeCBR, jCOLIBRI or eXiT*CBR, which will be briefly discussed in this section. FreeCBR[8] is a rather simple CBR engine, which allows the realization of basic CBR features. However, it does not cover features like case revision or retention and more individualized knowledge models, or comprehensive global and local similarity measures, are not applicable either. Further, it still requires quite some effort to apply it to a high variety of tasks. jCOLIBRI started from a task oriented framework also covering distributed reasoning [19], recently jCOLIBRI Studio [20] for more comprehensive support of building CBR knowledge has been introduced. Up to today jCOLIBRI includes more machine learning and semantic web features while myCBR focused on the knowledge required in the knowledge containers. Furthermore, creating individualized case representations and especially flexible similarity measures is the strength of myCBR. eXiT*CBR has also its roots in machine learning applications and is specialized for medical diagnosis tasks [16]. It has recently been extended in order to cope with more than one case base. In comparison to myCBR, the ideas behind the methodology also differ, since we are focusing on the knowledge container model rather than the machine-learning-related tasks. The integration of Drools in an existing framework for executing rules on a given corpus has been introduced by Hanft et al. [14]. In this paper Drools has been

[8] `http://freecbr.sourceforge.net/`

integrated in an existing OSGi environment. The approach presented here required a more comprehensive customization since myCBR was not embedded in OSGi and the requirements for the rules differed in terms of usable knowledge and modification of cases.

5 Summary

In this paper we presented myCBR 3 by introducing its core features as well as optional extensions for more complex applications. According to our Open Source strategy these extensions are also provided for myCBR users.

Our goal for the tool is to increase awareness of myCBR's capabilities and have more people use the tool. Further on, extensions such as the presented Information Extraction components are planned to be provided as add-ons. In these terms, we think of some kind of tool box or software product line [15] with the core myCBR framework that can be extended by, preferably Open Source, components on demand. The integration of Drools allows us to use completion and adaptation rules in applications, in addition we can also simulate constraints by the Drools rule mechanisms. However, currently myCBR does not support constraint-based problem solving.

The seven showcases presented have been implemented using myCBR GUI and SDK by different groups of users (students and myCBR developers). Currently we are collaborating with the University of West London, who are working on the explanation component of myCBR as introduced in SEASALT^{exp} [23].

The Information Extraction engine introduced in this paper enables an easier Knowledge Acquisition for CBR systems, because it supports a broad range of unstructured/weakly structured source data to well-formed heavily structured data and hence provides a scalable knowledge intensity. Because of the fact that many sources of experiential knowledge is User Generated Content, the presented approach makes this vast amount of unstructured data accessible for CBR applications more easily. The demonstrator applications introduced showcase the high variety of tasks myCBR can be applied to.

Acknowledgement. We would like to thank our students from the 2011 CBR lab course as well as Sara Marter, Julian Satzky, Fabian Schwahn and Irina Weizel for their great work and feedback on myCBR. Further, we thank the myCBR team members for their implementation efforts and valuable discussions.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications 1(7) (March 1994)
2. Adrian, B., Hees, J., van Elst, L., Dengel, A.: iDocument: Using Ontologies for Extracting and Annotating Information from Unstructured Text. In: Mertsching, B., Hund, M., Aziz, Z. (eds.) KI 2009. LNCS (LNAI), vol. 5803, pp. 249–256. Springer, Heidelberg (2009)

3. Althoff, K.-D., Bach, K., Deutsch, J.-O., Hanft, A., Mänz, J., Müller, T., Newo, R., Reichle, M., Schaaf, M., Weis, K.-H.: Collaborative multi-expert-systems – realizing knowlegde-product-lines with case factories and distributed learning systems. In: Baumeister, J., Seipel, D. (eds.) Workshop Proceedings on the 3rd Workshop on Knowledge Engineering and Software Engineering (KESE 2007) (September 2007)
4. Bach, K., Althoff, K.-D., Newo, R., Stahl, A.: A Case-Based Reasoning Approach for Providing Machine Diagnosis from Service Reports. In: Ram, A., Wiratunga, N. (eds.) ICCBR 2011. LNCS, vol. 6880, pp. 363–377. Springer, Heidelberg (2011)
5. Bach, K., Sauer, C.S.: mycbr student demonstrators at the computer cooking contest. In: Agudo, B.D., Cordier, A. (eds.) ICCBR 2011 Workshop Proceedings: Computer Cooking Contest Workshop (2011)
6. Bach, K., Sauer, C.S., Althoff, K.-D.: Deriving case base vocabulary from web community data. In: Marling, C. (ed.) ICCBR 2010 Workshop. Proc.: Workshop on Reasoning From Experiences on the Web, pp. 111–120 (2010)
7. Bali, M.: Drools JBoss Rules 5.0 Developer's Guide. Packt Publishing (2009)
8. Bellifemine, F., Caire, G., Greenwood, D.: Developing multi-agent systems with JADE. Wiley Series in Agent Technology (2007)
9. Bontcheva, K., Tablan, V., Maynard, D., Cunningham, H.: Evolving gate to meet new challenges in language engineering. Natural Language Engineering 10(3-4), 349–373 (2004)
10. Cunningham, H.: Information extraction, automatic. Encyclopedia of Language and Linguistics, 665–677 (2005)
11. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In: Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL 2002) (2002)
12. Endres-Niggemeyer, B., Jauris-Heipke, S., Pinsky, M., Ulbrich, U.: Wissen gewinnen durch Wissen: Ontologiebasierte Informationsextraktion. Information Wissenschaft und Praxis 57(6/7), 301 (2006)
13. Hanft, A., Ihle, N., Bach, K., Newo, R.: CookIIS – competing in the first computer cooking contest. Künstliche Intelligenz 23(1), 30–33 (2009)
14. Hanft, A., Schäfer, O., Althoff, K.-D.: Integration of drools into an osgi-based bpm-platform for cbr. In: Díaz-Agudo, B., Cordier, A. (eds.) ICCBR 2011 Workshop Proceedings: Process-Oriented CBR (2011)
15. van der Linden, F., Schmid, K., Rommes, E.: Software Product Lines in Action - The Best Industrial Practice in Product Line Engineering. Springer, Heidelberg (2007)
16. López, B., Pous, C., Gay, P., Pla, A., Sanz, J., Brunet, J.: exit*cbr: A framework for case-based medical diagnosis development and experimentation. Artif. Intell. Med. 51(2), 81–91 (2011)
17. Marter, S.: Case based coordination agents - knowledge modularization and knowledge composition (Fallbasierte Koordinationsagenten – Wissensmodularisierung und Wissenskomposition für dezentrale, heterogene Fallbasen). Master's thesis, Institute of Computer Science, University of Hildesheim (2011)
18. Newo, R., Bach, K., Hanft, A., Althoff, K.-D.: On-demand recipe processing based on cbr. In: Marling, C. (ed.) ICCBR 2010 Workshop Proceedings: Computer Cooking Contest Workshop, Karlsruhe, Germany, pp. 209–218 (July 2010)
19. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A.: A distributed cbr framework through semantic web services. In: Bramer, M., Coenen, F., Allen, T. (eds.) Research and Development in Intelligent Systems XXII (Proc. of AI 2005). pp. 88–101. Springer (December 2005)

20. Recio-García, J.A., Díaz-Agudo, B., González-Calero, P.A.: Template based design in colibri studio. In: Proceedings of the Process-oriented Case-Based Reasning Workshop at ICCBR 2011, pp. 101–110 (2011)
21. 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 (LNAI), vol. 5803, pp. 556–563. Springer, Heidelberg (2009)
22. Richter, M.M.: 1. Introduction. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (eds.) Case-Based Reasoning Technology. LNCS (LNAI), vol. 1400, pp. 1–15. Springer, Heidelberg (1998)
23. Bergmann, R.: 2. Experience management. In: Bergmann, R. (ed.) Experience Management. LNCS (LNAI), vol. 2432, pp. 25–44. Springer, Heidelberg (2002)
24. Roth-Berghofer, T.R.: Explanations and Case-Based Reasoning: Foundational Issues. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 389–403. Springer, Heidelberg (2004)
25. 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)