CookIIS Mobile: A Case-Based Reasoning Recipe Customizer for Android Phones

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

[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

http://www.dfki.de/web/competence/cccbr/

Abstract. Case-Based Reasoning (CBR) has been successfully applied for recommending recipes within different scenarios in the course of the Computer Cooking Contest and beyond. However, recipe recommendations are more useful when there are available in the kitchen, looking at the fridge and planning to cook a dish. In this paper we will present the *CookIIS* Android App, which has been implemented using *myCBR 3* and extended with the Open Source Business rule engine Drools. We will show how completion and adaptation rules can be processed with this new extension and how they affect the customization of cooking recipes.

Key words: Case-Based Reasoning, Android App, myCBR, Reuse, Adaptation and Completion Rules

1 Introduction

In this paper we will present a Case Study that uses the *myCBR 3* SDK [2] for the development of mobile applications as well as includes the Drools Rule Engine for providing case completion and adaptation. We will use the Computer Cooking Contest (CCC) prototype of our group to showcase how these two extensions can enhance *myCBR* applications.

CookIIS today is completely based on open-source software and it now made its way closer to the user's current situation since it has an Android App. It considers the restrictions given by the user and adapts available recipes to fit the user's wishes as much as possible. It provides cooking competence for those not so familiar with cooking and by this opens the world of cooking for an increasing number of people.

Customizing recipes can be challenging, because it is subjective. The taste of people selecting and modifying can not be generalized. However, mobile apps

provide many personal information of a user that can describe the context in which an intelligent software system can adopt to its user. The Android application for *CookIIS* is probably together with the insurance recommendation app presented by [7], one of the first mobile apps including Case-Based Reasoning (CBR). We believe CBR can be of a high value for mobile apps in various domains, because it supports creative solutions or recommendations right at hand. From our experience up to now, target domains for mobile CBR application are personalized decision making and recommendation such as traveling, e-Health or product recommendations. Furthermore, applying CBR in user interaction is also a topic to be addressed for mobile applications, as it has been discussed for web applications by [3].

1.1 Motivation

Including CBR in mobile application can help a knowledge-based system to obtain user information for improving its task. Especially the exploitation of contextual information can improve the application itself by increasing trust in the results such a system produces.

Contextual information of mobile devices can be either provided by sensors describing the current location, date and time, or the proximity of other devices and therewith people. Furthermore, from the content of mobile phones even more information - especially user characterization and preferences - can be obtained and incorporated in mobile apps.

For the home cooking domain, information such as nutrition preferences (diets, allergies, etc.) or available ingredients and tools are important. But also the season, that influences the availability of fruits or vegetables and the favor of certain ingredients, should be taken into account. Moreover we know that individual competences and experiences in cooking and the time available for preparation has a high influence on the meal and menu selection. From our point of view, the application domain of cooking is manifold and we are aiming to provide technology show cases, which can be adapted within other domains.

With the application presented in this paper we surely do not address all these challenges, however, we try to provide the information in place and show how it can be customized according various parameters.

In this paper we are addressing recipe recommendation and adaptation and the remaining of this paper is structured as follows: First we will give an introduction to *CookIIS* including its general concepts and the architecture. Section 3 describes the customization of recipes using rules, while section 4 describes an Android app and shows how the CBR features are supporting the user in finding recipes. The last sections summarizes the current status of *CookIIS* and gives an overview on the next steps.

2 CookIIS

CookIIS is a 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 in respect to allergies or dietary practices. *CookIIS* stores recipes and provides upon the user's request the most similar ones, adapts them to the user's wishes and by this fulfills specific constraints (allergies, vegetarian food etc.).

CookIIS is a CBR cooking recipe engine based on the open source tool *myCBR* that can be used via a web interface or Android app. It is based on an integrated knowledge model based on the former *CookIIS* model and makes use of the more formalized recipe base provided by WikiTaaable. The actual system now supports the full 4R CBR cycle [1]. Besides the similarity-based retrieval and the application of specific requirements (dietary practices, allergies, exclusion of certain ingredients), the system applies model-based adaptation alongside regular adaptation based on rules. For the processing of completion and adaptation rules we now use the Open Source rule engine Drools. *CookIIS* also collects user feedback (revise) and includes new cases semi-automatically (retain).

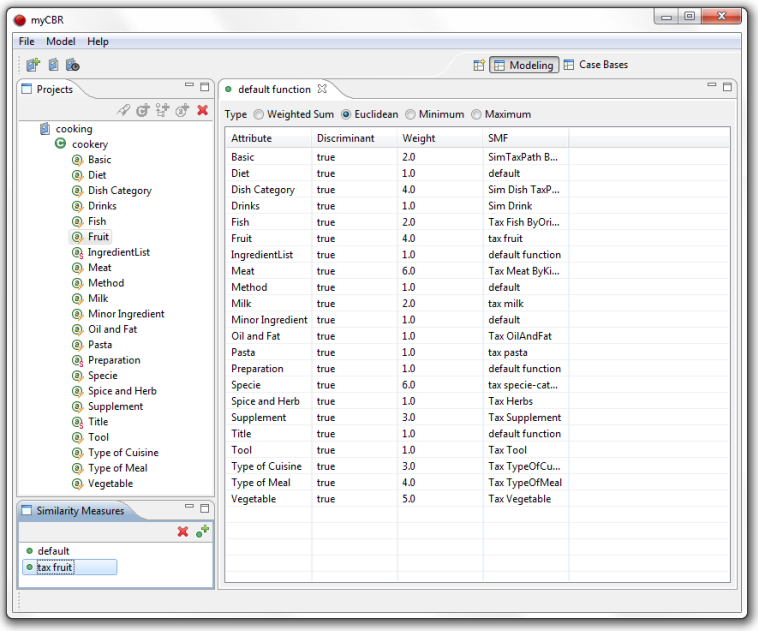

Fig. 1. CookIIS Knowledge Model in myCBR

CookIIS is based on a comprehensive case representation including 22 attribute descriptions of which 19 symbolic attribute descriptions represent the different kinds of ingredients, types of meals and cuisines. Three String attribute descriptions capture the title, preparation and raw ingredient data. The symbolic attribute descriptions currently contain 2,416 modeled attribute values.

The recipes included in the prototype originate in the CCC case base. The current case base contains the former Computer Cooking Contest (CCC) Compulsory case base with 1,484 recipes, which have been derived from the Wikitaaable system [4]. Based on the source data, we load the cases from a PostgreSQL data base into *myCBR*. The knowledge model along with the weights for each attribute description is shown in Figure 1.

Quality improvements have been carried out manually and automatically throughout the last 5 year through term, Bachelor's and Master's theses at the University of Hildesheim. The idea behind this is that incoming case data would be validated according to state of the art methodologies for experience management. Currently our knowledge model contains terms to index the CCC compulsory tasks' case base. If novel recipes are submitted through the web interface, they are not immediately loaded into the case base. Currently a knowledge engineer has to initiate the case loading process and, if necessary, include new attribute values. Currently we automatically verify the knowledge model against the case base.

2.1 System Architecture

CookIIS is based on a client-server-architecture, which allows us to provide both - a web-based and mobile application (see Figure 2).

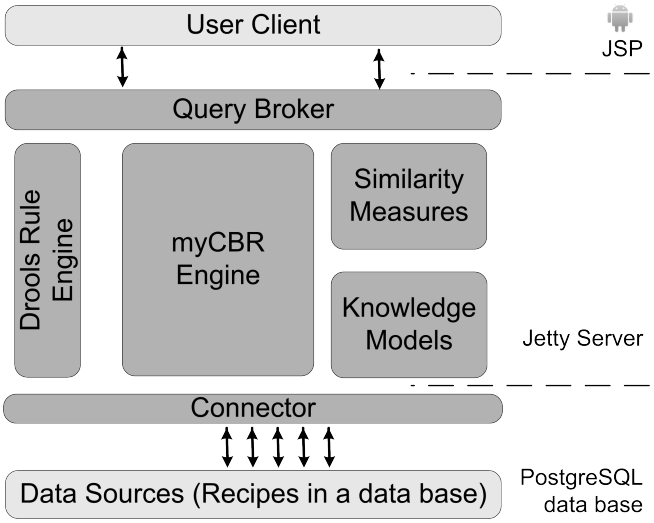

Fig. 2. CookIIS Architecture

The CBR engine, which is wrapped in the OSGi framework, is deployed on a Jetty Server. Further on, the cases and rules are mostly maintained in a PostgreSQL database which is accessed from the application to load cases and

create the rule set to be applied. The server itself uses servlets for exchanging data while the communication between the Server and the mobile application is implemented using JSON[3]. For the development of the Android application we have used the Android SDK.

The communication between client and server is carried out via the JavaScript Object Notation (JSON). JSON is a very minimalistic format based on the java JavaScript Notation for exchanging data between two communication parties. Compared to XML, JSON requires resources to represent data, which makes it an ideal match for mobile applications, because it is built using simple key-value-pairs (objects) and ordered value lists (arrays).

The following listing shows a query send to the *CookIIS* server. Each query contains the query type and the according specifications.

```
{"SEARCH: "Extended",
"Query": "chicken",
"Exclude": "garlic",
"Allergies": "nut",
"Diet": "low cholesterol",
"Rating": 10}
```

The query is then processed by the CBR Engine. Therefore first a new case has to be created which includes in the specifications. Afterwards the similarity-based retrieval is carried out. Since the specifies parameters that might require an adaptation, the rule engine is activated. Also the existing ratings are taken into account, which then influences the ranking. The following listing shows the retrieval result.

```
{"recipe_name":"Chicken in salsa verde;",
"adapted":true,
"newSim":0.30,
"recipe_categorie":"main course;",
"recipe_ingredients_list":[{"Oil and Fat":"oil"}, {"Meat":"
    chicken"}, {"Vegetable":"onion", "red pepper", "tomato", "
    green chilli",}, {"Fruit":"apple juice"}, {"Spice and
    Herb":"cilantro"}],
"recipe_avg_rating":4,
"dislike_counter":0,
"recipe_preparation":"Spray a non-stick skillet with vegetable
     cooking spray. Heat oil and saute chicken. Add onion and
    <font color='#D3D3D3'>(garlic)<\/font> <b>red pepper<\/b>
    , stirring until limp. Add remaining ingredients. Reduce
    heat, cover, and simmer about 15 minutes.",
"recipe_id":"cookery881",
"oldSim":0.34,
"recipe_rating_count":2,
"recipe_ingredients":"2 ts Canola oil [...]"}
```

[3] http://json-lib.sourceforge.net/index.html

The advantages of JSON in our scenario are that it can be parsed using less resources and it is independent from the programming language. Having a look in line 3 and 10 one can see that the adaptation and the rating weight caused a decrease of the similarity assigned to that case. Further on, line 5 the classification of the ingredients found while line 8 already contains HTML markup for displaying the adaptation's result to the user.

3 Recipe Customization

The customization of recipes is done within the reuse step of the 4R cycle by so called adaptation rules. The core of *myCBR* does not contain any support for rules and for that reason we were investigating various rule engines that could be integrated.

JBoss Drools[4] has been applied as rule engine for adaptation knowledge in another case study [6] and we decided to evaluate it for *myCBR* as well. Eventually we decided to use Drools rather than the Open Source and Java-based rule engines jRete[5] or jRuleEngine[6], because Drools comes with a clean and modularized structure that allows us using the functionalities we need. Further on there are tools for maintaining, debugging and editing rules.

Drools in general is an Open Source Framework for developing rule-based systems. The framework contains tools for creating, compiling and executing rules. It requires that first *IF fact(s) THEN action* rules have to be created. The facts are extracted from the case to be adapted and the actions are applied to that particular case. The facts are stored in memory, so an inference engine can recognize the given pattern in order to apply the desired actions. Within *CookIIS*, these rules either enhance the case description or substitute ingredients.

In Drools, rules are defined and stored in a particular language called Drools Rule Language (DLR). The major functionalities are available as bundle and service, so they are easy to integrate in an existing OSGi framework. *myCBR* itself had also to be integrated in its own bundle now providing case retrieval services. Therefore each method can be used to create, change and retrieve cases has to be provided. With the current implementation, we provide general CBR services, so the OSGi infrastructure of *myCBR* can be reused. Of course, changes in the SDK's API, would also require OSGi refinements.

In *CookIIS* three different types of rules are implemented: completion, adaptation and exclusion rules. These rules are provided as drl-files, which can either only handle the IF-THEN clauses or more comprehensive tasks as described in section 3.1.

Extensions that Drools did not directly support are the handling of knowledge models (in form of ontologies or taxonomies) in general and the modification of attribute values in cases. This functionality has been implemented as a service of the rule engine, which can be activated and deactivated during runtime. Even

[4] `http://json-lib.sourceforge.net/index.html`
[5] `http://sourceforge.net/projects/jrete/`
[6] `http://jruleengine.sourceforge.net/`

if we are currently not using this feature from the application, we can imagine that this dynamic activation can increase the customization.

Completion Rules The process of applying Drools rules as completion rules is pictured in Figure 3. After the initial case base has been loaded, the completion rules are loaded from a CSV file and executed on every case. They enhance the cases with information based on the existing attribute values.

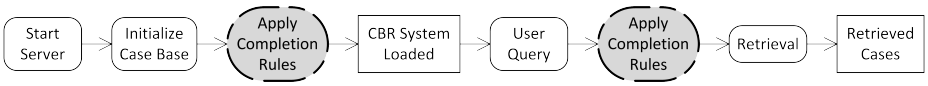

Fig. 3. Completion Rules for *myCBR* based on Drools

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. An example for such a rule is

IF case.title = Kippferl THEN *set* case.typeofcuisine = austrain

In the same manner, the rule engine applies completion rules to each query submitted to the system.

Adaptation Rules Adaptation rules are more complex since the conditions as well as their following actions might cause time-consuming changes on a case. These rule types are used to either remove or replace unwanted ingredients and is therewith one of the core features of the case customization. They are depending on the underlying knowledge models such as similarity tables, taxonomies, calculations, adequate substitution candidates. Also 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.

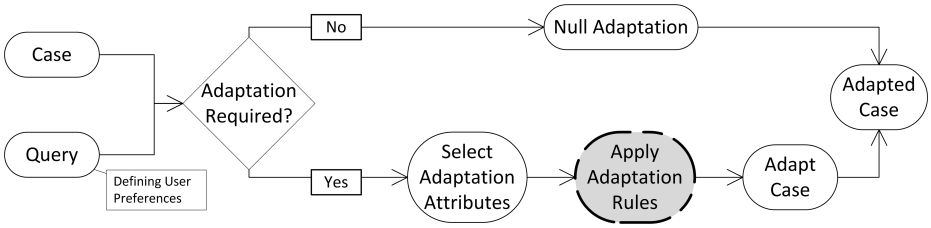

Fig. 4. Adaptation Rules for *myCBR* based on Drools

Figure 4 shows that we first check whether an adaptation is necessary, before we execute the rules on a small subset of cases. For each case the attribute description that will be adapted is selected and the rules are applied.

3.1 Model-based Adaptation

The model-based adaptation has been described in [5]. We kept the basic methodology, but transfered it into the *myCBR*-Drools Framework. This kind of adaption produces reliable, but less creative substitutions, because substitution candidates are retrieved by browsing common subsets.

3.2 Community(-Driven) Adaptation

The basis for the newly integrated adaptation rule set are experiences provided within a cooking community. Therefore we have analyzed comments of a large German cooking portal. Each comment provided by users for a recipe has been crawled and mined whether it includes adaptation knowledge for that recipe. In particular, we applied Text Mining for identifying terms describing ingredients using our knowledge model. Furthermore, the previous version of *CookIIS*, which was based on the empolis Information Access Suite, contained the ingredient names in German and English, which allowed us to translate the recognized terms easily.

After we found terms indicating ingredients, we compared the recognized ingredients with those included in the original recipe. Matching ingredients were additionally tagged as $< old >$ and the remaining as $< new >$. In the next step, we searched for signal words that point to a substitution such as *instead*, *replace*, etc. and tagged them - including them as well. In the final step, we were looking for sets of $< new >< signal - word >< old >$, which are close to each other and extracted them as rules.

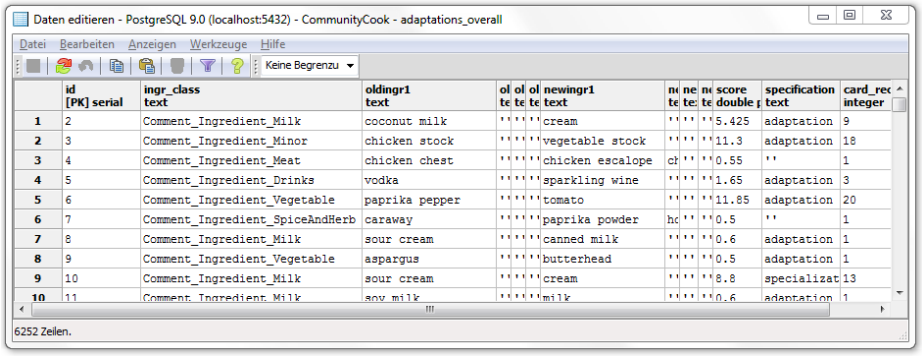

Fig. 5. Community Adaptation Knowledge (raw)

To increase confidence rules, we also extracted user ratings and information whether users found a certain comment helpful. This mainly removed all the *yummy*, *delicious dish* and *thanks* comments. After extracting these pairs, they are stored in the PostgreSQL database as it can be seen in Figure 5.

As you can see, we are also capturing more than one ingredient, but the quality of these substitutions was not sufficient, so we only included substitutions with clear ingredient assignments and a high confidence score.

3.3 Exclusion Rules

Exclusion rules are more strict than adaptation rules, because they are removing results from the result set, if given conditions match. We decided to provide these functionality to better support people with allergies and give the user the opportunity the manually exclude ingredients. Since those rules do not need any time of inference, we implemented them within the application rather than using Drools. The input for the exclusion rules is retrieved from the user. After the retrieval has been carried out, recipes matching the exclusion rules' condition, they are removed from the result set.

4 CookIIS Android App

The mobile app, which is one of the interfaces of *CookIIS* provides a search interface that lets the user specify which kinds of ingredients are available and which should be avoided. Figure 6 only shows the simple search interface. The extended search also allows to specify diets, allergies and the weighting of user comments.

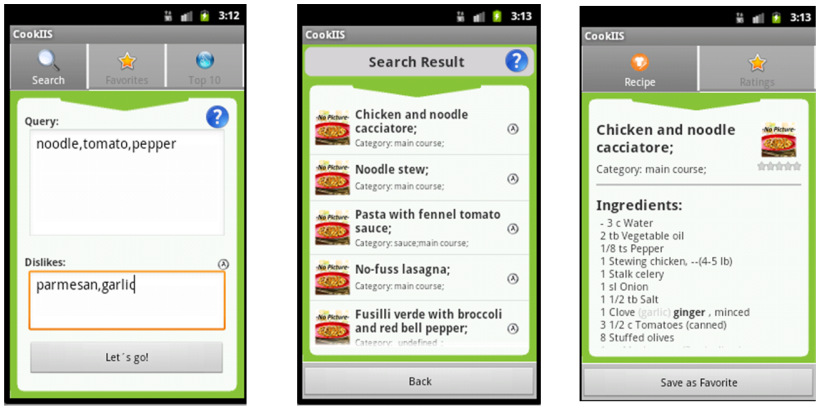

Fig. 6. CookIIS Android app showing the query (left), the result list (middle) and the recipe details including an adaption (right).

After the query has been specified, a list of ten recipes is presented to the user from which he can choose. The 'A' on the right of a recipe indicated that this has been adapted. When the user selects one of the recipes, the ingredient list and preparation instructions are shown. Furthermore, adapted ingredients (such as garlic vs. ginger in Figure 6) are marked. Further on, recipes can be rated, add to the users favorite list and the ingredients can be added to a shopping list. The user's favorites are used to recommend new recipes from the recipe base. Based on the ingredients often used recipes are suggested. The shopping list allows the user to mark available ingredients and also send those that are not available via text message to someone from his/her contact list[7].

4.1 Discussion

From our experience up to know, including the different kinds of adaptation rules improves the retrieval results. However, the suggestions proposed by the model-based adaptation highly depend on the quality of the knowledge model. Further on, currently only a random ingredient is proposed to be the substitution candidate, because we do not maintain further information on successful adaptations. On the other hand, the user interface should contain only the information necessary and we should not try to receive feedback at every possible step.

During a short-term evaluation, which we placed underneath the retrieval results on the website[8], we found out that most of the users were satisfied with the results. Figure 7 shows the user ratings.

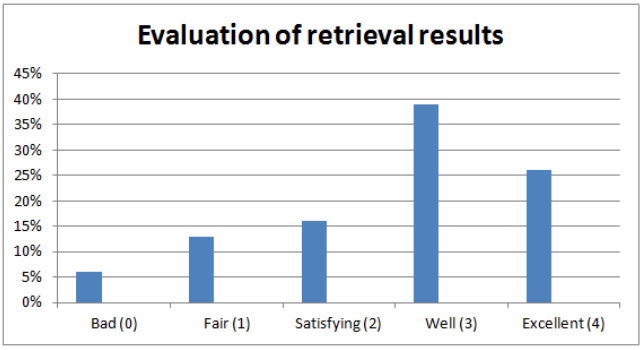

Fig. 7. Retrieval Result Evaluation

[7] A short video of how the app looks like and runs can be found at `http://www.youtube.com/watch?v=BgiPtpdYapU&feature=plcp`

[8] `http://grievous.iis.uni-hildesheim.de:8080/myCookies/WebContent/JSPs/index.jsp`

When looking through the results it seemed that substitutions for spices and herbs are usually rated quite poor, while fruits and vegetables are evaluated much better. For that reason, we think about applying different adaptation strategies on the attribute description. An idea in this context would be to only use community-based adaptation rules for spices rather than the model-based.

Most of the work presented here, was carried out by students as their term or bachelor's thesis, which shows, that both, *myCBR* and Drools, can be handled with relative easy and the success shows up quickly.

5 Summary and Outlook

In this paper we described how we have extended the *myCBR 3* SDK in order to use various types of adaptation and completion rules. The rule engine we have integrated is JBoss Drools, which required to build an OSGi framework around the *myCBR* API, but it allowed us to access the *myCBR* knowledge models and carry out a model-based adaptation and directly adapt cases without any workarounds.

A goal for the future is to create adaptation cases rather than adaptation rules to describe more precisely how cases can be adapted. We think making use of user generated content from an active community brings in many advantages: First of all the amount of available data allows a more customized adaptation with regard to a users preferences. Further on, creating a user context based on the information provided by a mobile device (such as location, time frame, season, activities), the adaptation can be even more personalized.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning : Foundational issues, methodological variations, and system approaches. AI Communications 1(7) (Mar 1994)
2. Bach, K., Althoff, K.D.: Developing Case-Based Reasoning Applications Using myCBR 3. In: Watson, I., Agudo, B.D. (eds.) Case-based Reasoning in Research and Development, Proceedings of the 20th International Conference on Case-Based Reasoning (ICCBR-12). pp. 17–31. LNAI 7466, Springer (September 2012)
3. Bridge, D., Healy, P.: Ghostwriter-2.0: Product reviews with case-based support. In: Bramer, M., Petridis, M., Hopgood, A. (eds.) Research and Development in Intelligent Systems XXVII (Procs. of AI-2010, The Thirtieth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence). pp. 467–480. Springer (2010)
4. Cordier, A., Lieber, J., Molli, P., Nauer, E., Skaf-Molli, H., Toussaint, Y.: WIKITAAABLE: A semantic wiki as a blackboard for a textual case-based reasoning system. In: SemWiki 2009 - 4rd Semantic Wiki Workshop at the 6th European Semantic Web Conference - ESWC 2009. Heraklion, Grèce (May 2009)
5. Hanft, A., Newo, R., Bach, K., Ihle, N., Althoff, K.D.: Cookiis - a successful recipe advisor and menu advisor. In: Montani, S., Jain, L. (eds.) Successful Case-based Reasoning applications, p. to appear. Springer (2010)

6. Hanft, A., Schäfer, O., Althoff, K.D.: Integration of drools into an osgi-based bpm-platform for cbr. In: Agudo, B.D., Cordier, A. (eds.) ICCBR-2011 Workshop Proceedings: Process-Oriented CBR (2011)
7. Sauer, C.S., Hundt, A., Roth-Berghofer, T.: Explanation-Aware Design of Mobile myCBR-Based Applications. In: Watson, I., Agudo, B.D. (eds.) Case-based Reasoning in Research and Development, Proceedings of the 20th International Conference on Case-Based Reasoning (ICCBR-12). pp. 399–413. LNAI 7466, Springer (September 2012)