Integration of Drools into an OSGI-based BPM-Platform for CBR

Alexandre Hanft[1], Oliver Schäfer[1], and Klaus-Dieter Althoff[1,2]

[1] University of Hildesheim
Institute of Computer Science – Intelligent Information Systems Lab
Marienburger Platz 22, 31141 Hildesheim, Germany
{firstname.surname}@uni-hildesheim.de

[2] Competence Center Case-Based Reasoning
German Research Center for Artificial Intelligence (DFKI) GmbH
Trippstadter Strasse 122, 67663 Kaiserslautern, Germany

Abstract. This paper describes the integration of the business rule management system Drools inside the framework SMILA which is a workflow system. The goal behind that effort is to foster the adaptation with rules and completion rules not only for CBR-systems, but in business context. Existing open source CBR systems have none or very limited adaptation capabilities as well as proprietary, commercial systems restrict complex adaptation processes. Our so-called RACK implementation brings the power of Drools into SMILA and makes the experience of already written rules in business area available.

1 Introduction

Existing open source CBR systems have unfortunately none or very limited adaptation capabilities. For instance the popular *JColibri* currently only allows to change certain fixed values or a relative modification for numerical values [1]. Another popular open source system for CBR – *myCBR* – evolved from a plug-in of Protégé to an Eclipse RCP-based application in its upcoming version [2] and has a powerful similarity modelling, but no support for adaptation or completion rules. Unfortunately, both does not take workflow concepts explicitly into account.

On the other hand, Attensity Research & Discovery [3], (formerly e:IAS), one of the most popular industrial strength CBR-capable tool, has a (proprietary) pipelining concept to organise tasks in workflows and a powerful rule engine for completion of queries and adaptation of cases. But this tool is limited regarding capabilities for complex adaptation processes. For instance the rule engine (performing the adaptation) has no direct access on similarity measures and values, no full access on the case/ query. Furthermore, its concept of handling values with the concept of set theory lacks in the possibility to iterate over each item inside the set (details in [4]). Some of the limitations can bypassed by self-programmed pipelets, but using their API requires a lot of internal knowledge.

To overcome limitations of proprietary software despite others problems like scalability, in 2008 a new framework *SMILA* was proposed, the SeMantic Information Logistics Architecture. SMILA has a clear focus on business enterprises and uses an established workflow mechanism, BPEL pipelines. SMILA is designed with a very general purpose and allows to use several search technologies, but its open source edition (the only we consider) lacks some capabilities for CBR, at least similarity-based retrieval and adaptation. Indeed, Apache Lucene used for retrieval at the moment.

Hence, we detect a gap between classic CBR systems with rule-based Adaptation and business-oriented rule-based systems as well as workflow systems we aim to bridge with our approach. What we found necessary is a workflow-oriented tool that performs CBR with a strong adaptation in the enterprise context to use their already existing (business-rule) experience. Our medium term goal is to provide a workflow of several adaptation processes, not only in the CBR area but in the business environment. The necessity for a workflow of adaptation follows also from the experience with the Improvements of the Model-based Adaptation in CookIIS [5, p. 207]. Consequently our aim is now enrich the SMILA framework with CBR capability. Beside the retrieval, we focus here on the workflow and a rule engine to achieve a powerful adaptation.

JBoss Drools is a common, high-performance open source business rules tool with proven quality [6]. It allows inside the rules full object access on self written Java classes (beans), calling of own functions for arbitrary functionality, debugging etc. Additionally, Drools proposes (in contrast to other rule-engines) to be OSGi-ready which recommends it for integrating in other OSGi-based systems. Due to effort and the afore mentioned advantages we decided us for the integration of Drools into SMILA instead of an own implementation of a rule engine.

The next section 2 gives an overview of the Frameworks SMILA and Drools. We proceed in section 3 with the concept and implementation of the integration of Drools into SMILA, called RACK. Afterwards, in section 4 we present some details of the evaluation of the framework and describe the applicability of RACK for CBR. We finish with a conclusion, discussing related work and give an outlook on future work.

2 Used Frameworks: SMILA and Drools

2.1 SMILA

The framework *SMILA* (SeMantic Information Logistics Architecture) is an Eclipse RT project [7] aimed for building Java Applications to index, process and search unstructured informations. Its goal was to overcome the disadvantages of their predecessors Research & Discovery/e:IAS. SMILA bases in contrast to the proprietary model of e:IAS on standards technologies like OSGi, BPEL, JMX, SCA and SDA to easier the maintenance and integration of third-party components.

The OSGi Service Platform is a specification of the OSGi Alliance which provides an API to build modular Java-Applications. The main components of an OSGi based Application are bundles and services. These can be installed, started, stopped, updated and de-installed at the Service Platform at runtime to dynamically exchange functionality at system runtime.

SMILA organises its processes in a workflow model running several BPEL pipelines which are managed by a workflow engine. These are at least *preprocessing* and *information retrieval*. Figure 1 shows an overview of the architecture of SMILA [8]. On the left hand side the components for *preprocessing* can be seen as well as the components for *information retrieval* on the right hand side. A workflow here is called pipeline. Single activities in such a pipeline are called pipelets representing reusable Java Components [9]. The communication between these pipelets is managed by a pipeline- and invocation-specific blackboard. This concept is very similar to them of e:IAS, but now it does not need internal knowledge to configure or implement own components, because its specification is open and well-known in literature.

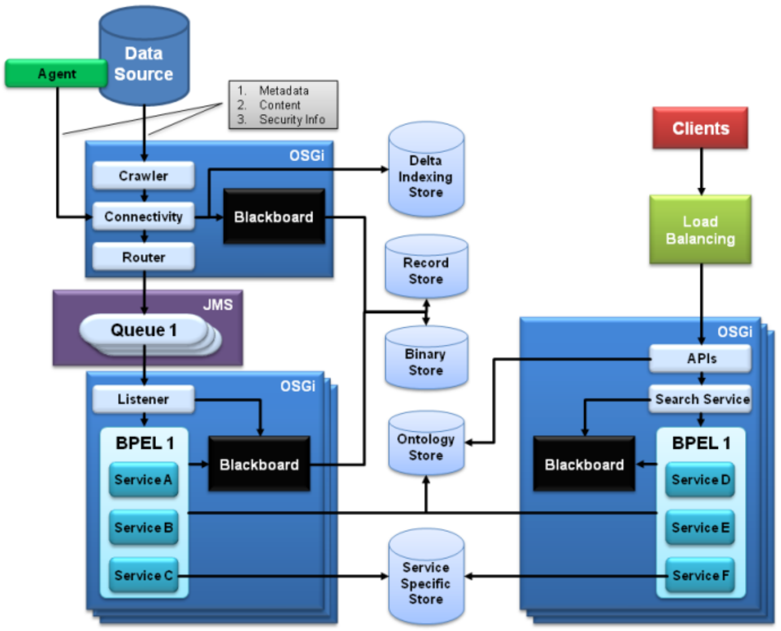


Fig. 1. Overview of the SMILA framework [8]

2.2 Drools

The "Business Logic integration Platform" *Drools* is one of the most common open source tools for rule-based systems, workflow and event processing [10,6]. This platform consist of five parts, but we concentrate here on the core component: Drools Expert (the rule engine). Drools Expert performs forward chaining inference with an enhanced version of the Rete algorithm.

Figure 2 shows the key components of Drools Expert. The `KnowledgeBuilder` is used to build `KnowledgePackages` representing the knowledge in form of rules or fact types. For instance it take a drl file, the Drools standard format for rules, analyses it and compiles it into a knowledge package. One or more knowledge packages build a `Knowledge Bases`. For each knowledge base it is possible to initialise a `Knowledge Session` which represents it at runtime. After the initialisation of a session facts can be inserted and a set of commands can be executed. The most important command is the `FireAllRuleCommand`. A knowledge session can be stateless or stateful. Only the latter saves the state of the facts between multiple executions. However, a stateful session has to be disposed explicitly, while a stateless session will be disposed automatically after each execution [6].

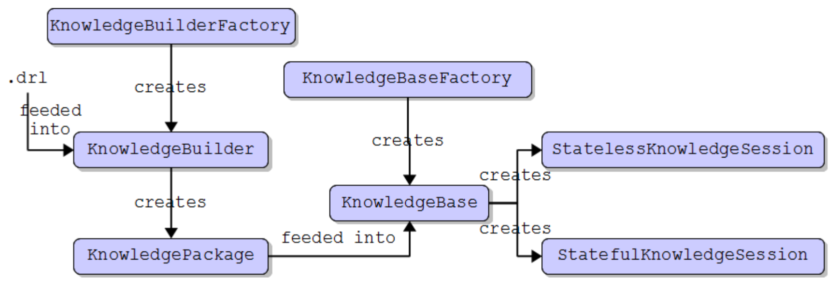


Fig. 2. The process of creating knowledge sessions, following [6, p. 22]

3 Design and Implementation of RACK – the Drools integration

This section describes the concept of the proposed integration as well the behaviour of the components during the workflow is running. The implementation *RACK* stands for *Rule-based Adaptation of Case-based Knowledge* which represents the goal of the integration to make SMILA CBR-ready.

Figure 3 shows the structure of the components (OSGi bundles) which had to be integrated. Based on the existing Drools bundles, in particular for the initialisation of knowledge bases and knowledge sessions, they should be used by the SMILA bundles. For this purpose it was necessary to implement connection bundles (in the figure as Drools-SMILA, later RACK). To process the records, for instance firing rules, the integration in the BPEL pipelining concept of SMILA was focused. For this reason some services and pipelets were implemented providing the access and execution of the Drools Expert components.

Moreover, it was also necessary to handle the needs of RACK users who act as fact and rules supplier. To enhance the usability and scalability it was also important that the consumer can extend the basic integration components by own classes for their own domain and business-logic. Domain-specific processing cannot be handled via general pipelets and services. Hence, abstract pipelets (AP) and helper classes had been implemented. These and the basic pipelets can be used via derivation to implement domain-specific pipelets.

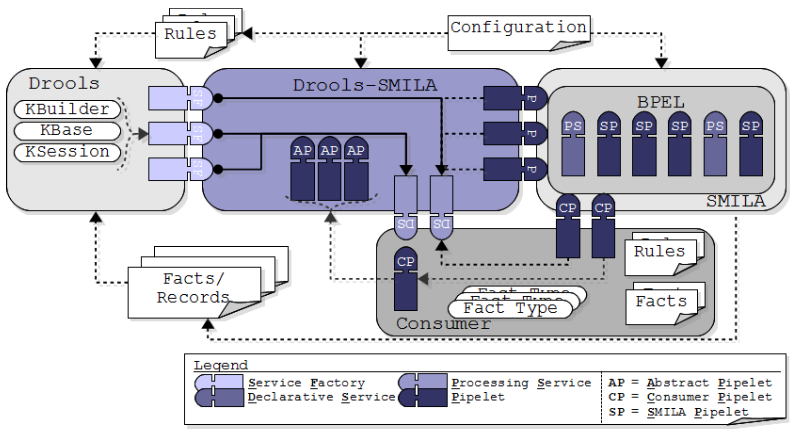


Fig. 3. The conception of the integration RACK together with its bundles, files, services and pipelets [11]

3.1 The Workflow – running the services in SMILA

As shown in figure 4 the processing of an user request (cf. query) (step (1)) is performed through a sequence of pipelets and processing services in a BPEL pipeline. The initial search service (LuceneService) pushes records (cf. cases) according to the query on the blackboard (2). Followed by initialising the session (3), these records can be pulled from the blackboard (4) and processed by the following pipelets and services. The RACK pipelets use the blackboard not only to load records (4), but in the knowledge sessions they also temporary save a session (6) and request active SessionID (7) or objects like execution results. Other pipelets or services can reuse a session by requesting it from the SessionService via its SessionID (8). Finally each pipelet can execute the session (5,9). The steps 7–9 are only necessary for reusing a session. For a stateful session it is necessary to dispose it after the whole processing is done (10).

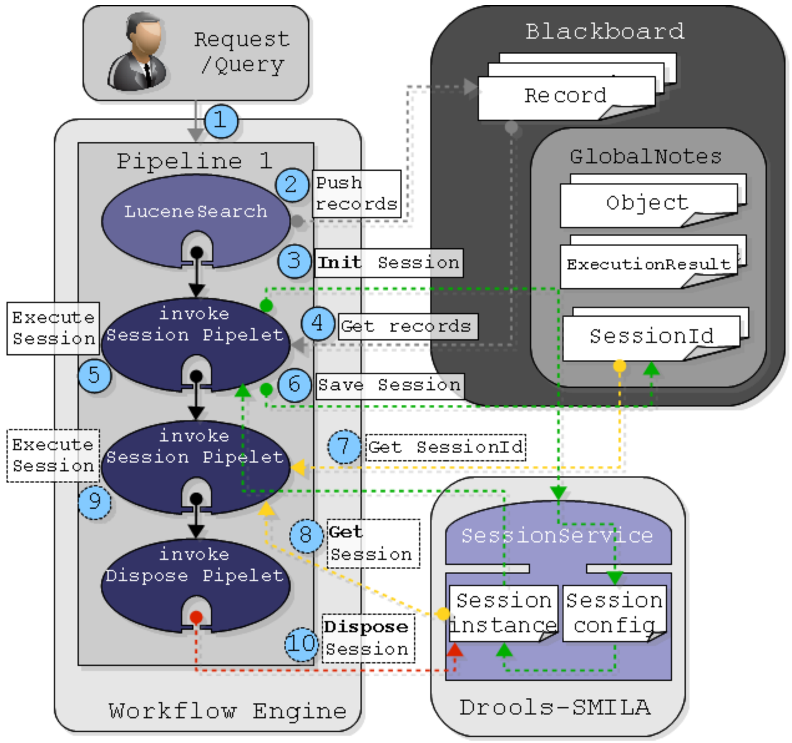


Fig. 4. Record processing in a BPEL pipeline inside RACK. [11]

The RACK implementation includes three bundles for the interfaces, the implementation itself and a bundle containing some use cases and examples. Moreover, RACK offers two declarative (OSGi) services for managing the session and knowledge base instances. They are not accessible within the BPEL pipelines and act like a background worker. The processing in the BPEL pipelines can be performed through the following three basic pipelets and extended consumer-pipelets:

– `PipeletBatchExecutionStateless`: Initialises and executes a stateless knowledge session.
– `PipeletBatchExecutionStateful`: Initialises or reuses a stateful knowledge session and executes it.
– `PipeletDisposeStatefulSession`: Disposes a stateful session.

A pipelet can be configured separately for each pipeline invocation and processes a configurable list of commands. These commands can also be configured with subcommands to initialize or reuse existing objects. In short they are used to insert objects into a session or to trigger firing the rules.

There are a lot of other components in RACK, for instance for the configuration, more details can be found in [11]. Figure 4 points to another benefit from using Drools (or other rule systems) in a BPEL pipeline, the component behaves like all other pipelets operating on the records. As most important it is possible to perform other record adaptations between these invocations. This is especially necessary for complex adaptation processes which are composed from more than one adaptation approach performed sequentially (as three in [5]). For instance a pipelet have to select an adaptation suggestion from a database or community before the rule-based adaptation is called.

To reuse a stateful session, it is sufficient informing the session which records have changed instead of re-insert all records. This could be achieved by collecting the ID's of changed records on the blackboard, a flag inside the record or a publish-subscriber mechanism. Afterwards, facts and rules have only to be re-evaluated and the rules can fire again. A complying workflow looks like this:

1. Fire completion rules (PipeletBatchExecutionStateless)
2. Retrieve cases (LuceneService)
3. Adapting retrieved cases with rules (RACK) (PipeletBatchExecutionStateful), which corresponds to step 3–6 in Figure 4
4. possibly perform other adaptation approaches (third-party components)
5. possibly inform the stateful session which records are changed and
6. Re-evaluate the rules and fire them again (PipeletBatchExecutionStateful) corresponding to step 7–9 in Figure 4
7. Dipose the session (PipeletDisposeStatefulSession)

The steps 3, 4 and 5+6 can freely repeated – the underlying BPEL pipeline allows enable a complex adaptation designed as a workflow.

Using Drools has some other practical beneficial features. It is possible to prioritize the firing of rules via attributes (salience) or partition the *Agenda* of the rule base in groups (agenda-group). Therefore, only parts of the rule set could be fired, which supports the application of several different rule-based adaptations. While executing a knowledge session the focused Agenda can be changed at runtime, which implies changing the control flow at runtime towards a dynamic workflow.

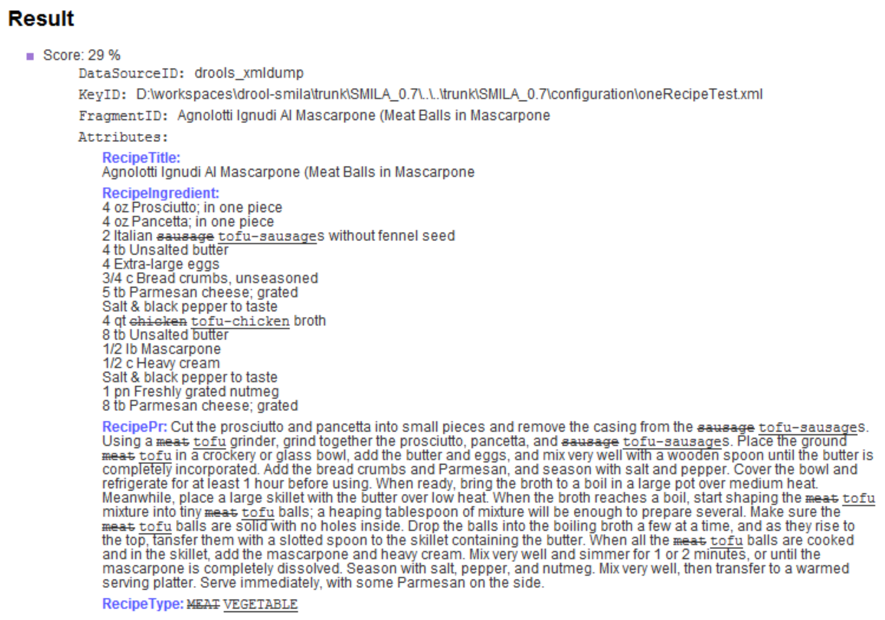
4 Evaluation and Applicability of CBR

4.1 Evaluation within the Cooking Contest Domain

The RACK implementation was evaluated by a simplified version of the CookIIS's model-based adaptation. The idea behind it is to replace forbidden ingredients (here meat) in the retrieved recipes by vegetarian alternatives. During the indexation each recipe (cf. record/case) of the Computer Cooking Contest's recipe base is categorised as *meat* or *vegetable* through completion rules. The users can query for a recipe with a specific name, ingredient and category (meat, vegetable, unknown). During the processing of the query completion rules are fired to resolve conflicts between given ingredients and category. Additionally, this categorization improves the retrieval by the `LuceneSearchService`.

```
when
  eval(global1_query != null) //(Query)RecipeType has not to be empty
  ... //definition of variables
then
  List<Literal> ingredients = $RecipeIngredients.getLiterals();
  Iterator<Literal> i = ingredients.iterator();
  while(i.hasNext()){
    Literal ingredient = i.next();
    if(Meat.isMeatConcept( ingredient.getStringValue())){//ingr. is meat?
      String[] sAdaption = Meat.findMeatAlternative(ingredient.getStringValue());
      ingredient.setStringValue( sAdaption[0]); //set ingr. to found alternative
      Annotation an = $record.getFactory().createAnnotation();
      an.addAnonValue( sAdaption[1] ); //add annotation with replaced ingr.
      ingredient.addAnnotation("adaption", an); } ... } ...
end
```

Fig. 5. excerpt from the Drools adaptation rule exchanging meat ingredients


Result

- Score: 29 %

DataSourceID: drools_xmldump
KeyID: D:\workspaces\drool-smila\trunk\SMILA_0.7\..\..\trunk\SMILA_0.7\configuration\oneRecipeTest.xml
FragmentID: Agnolotti Ignudi Al Mascarpone (Meat Balls in Mascarpone
Attributes:

RecipeTitle:
Agnolotti Ignudi Al Mascarpone (Meat Balls in Mascarpone

RecipeIngredient:
4 oz Prosciutto; in one piece
4 oz Pancetta; in one piece
2 Italian ~~sausage~~ tofu-sausages without fennel seed
4 tb Unsalted butter
4 Extra-large eggs
3/4 c Bread crumbs, unseasoned
5 tb Parmesan cheese; grated
Salt & black pepper to taste
4 qt ~~chicken~~ tofu-chicken broth
8 tb Unsalted butter
1/2 lb Mascarpone
1/2 c Heavy cream
Salt & black pepper to taste
1 pn Freshly grated nutmeg
8 tb Parmesan cheese; grated

RecipePr: Cut the prosciutto and pancetta into small pieces and remove the casing from the ~~sausage~~ tofu-sausages. Using a ~~meat~~ tofu grinder, grind together the prosciutto, pancetta, and ~~sausage~~ tofu-sausages. Place the ground ~~meat~~ tofu in a crockery or glass bowl, add the butter and eggs, and mix very well with a wooden spoon until the butter is completely incorporated. Add the bread crumbs and Parmesan, and season with salt and pepper. Cover the bowl and refrigerate for at least 1 hour before using. When ready, bring the broth to a boil in a large pot over medium heat. Meanwhile, place a large skillet with the butter over low heat. When the broth reaches a boil, start shaping the ~~meat~~ tofu mixture into tiny ~~meat~~ tofu balls; a heaping tablespoon of mixture will be enough to prepare several. Make sure the ~~meat~~ tofu balls are solid with no holes inside. Drop the balls into the boiling broth a few at a time, and as they rise to the top, tansfer them with a slotted spoon to the skillet containing the butter. When all the ~~meat~~ tofu balls are cooked and in the skillet, add the mascarpone and heavy cream. Mix very well and simmer for 1 or 2 minutes, or until the mascarpone is completely dissolved. Season with salt, pepper, and nutmeg. Mix very well, then transfer to a warmed serving platter. Serve immediately, with some Parmesan on the side.

RecipeType: ~~MEAT~~ VEGETABLE

Fig. 6. An adapted meat recipe

Figure 5 shows an excerpt of the main adaptation rule, which fires for each non-empty query and if the queried category is vegetarian and in conflict with the category of a retrieved recipe (meat). It iterates over each ingredient and if an ingredient is meat, an alternative is found (method findMeatAlternative() from own class meat). Afterwards this ingredient is set to its alternative and an annotation is added (more details in [11]). Figure 6 shows this result in SMILA for a query for vegetarian recipes after the rule-based adaptation. The adaptation rule identifies in the recipe *Agnolotti Ignudi Al Mascarpone (Meat Balls in Mascarpone) chicken* as a meat ingredient and replaces it with *tofu-chicken*. The model and rules in this evaluation prototype are not so complex as in CookIIS, but demonstrate the execution of Drools rules in SMILA and the full object access on the records (cf. recipe). Furthermore it would be possible to make use of ontologies via the SMILA integration of *SESAME* to extend the limited model.

4.2 Applicability of CBR

The concepts and components of the more general SMILA can be matched on CBR-specific concepts to provide CBR functionality with SMILA. A case is clearly a record of SMILA, with the one difference that problem and solution part are not strictly divided.Accordingly, the case base is the record store. The used search service Lucene performs a full text search instead of similarity based retrieval, but replacement is possible. Adaptation rules (as well as completion rules) can be realised through the aforementioned RACK implementation and perform the Reuse step from CBR. Revise is at the moment not possible with this implementation, but due to the workflow concept relatively easy to integrate. Retain is feasible in a rudimentary way by storing the changed records. What remains is to replace the search engine with one based on similarity measures as the next step towards a full-featured CBR system based in SMILA.

5 Conclusion, Related Work and Outlook

In this paper we presented *RACK*, a *R*ule-based *a*daptation of *C*ase-based *K*nowledge, implementing the integration of Drools into the workflow-oriented SMILA framework. It enables rule-based Adaptation in SMILA and is a great step forward to prepare SMILA as full-featured open source CBR system. This was integrated into the pipelining concept of SMILA based on the OSGi services, which facilitates a flexible orchestration of several subsequent adaptation approaches (as in [5] with e:IAS) through a workflow.

We gave an overview on the structure of bundles RACK consists of and on the different possibilities to implement regarding sessions. Afterwards we described exactly how the sequence of the called services in SMILA works during lifetime, which is to be more specific, during the execution of a BPEL pipeline "information retrieval" after the initiating query of an user or agent.

After a successful evaluation showed its general availability for use we described the mapping and applicability of RACK with SMILA for CBR. According to the general focus of SMILA RACK is not restricted to adaptation inside CBR, but explicitly designed to fulfil the specific requirements of CBR systems.

5.1 Related Work

SMILA is the official successor of Research & Discovery/e:IAS. From a broad perspective the rule engine inside e:IAS achieves the same goal as the Drools integration in SMILA, but it is open source and have a lot of features missed there as full object access on self written Java classes, own functions for arbitrary functionality, debugging etc. Some open source CBR systems are myCBR[2], JColibri[1] or IUCBRF[12]. Integrating a rule system into existing open source CBR frameworks is fairly imaginable, but we prefer to move our focus to a more general and business context. Our RACK facilitates rule-bases adaptation from CBR in the context of workflow-oriented business applications in contrast to the adaptation of a workflow with the aid of CBR, as for instance in [13].

5.2 Outlook

The evaluation was done with stateless sessions. Therefore, we will build a second prototype with stateful sessions and investigate the interactions between other adaptation approaches and our rule-based adaptation in workflows. We plan further implementations to enhance SMILA as a full-fledged CBR system framework. Therefore we looking for a similarity-based retrieval service, we will try to reuse a retrieval component equipped with a web service interface from former projects. Moreover, Drools can use DSLs as special rule dialects to facilitates simpler, human-centered adaptation languages. Due to other development activities on myCBR we aim at repeating the integration of Drools for myCBR to provide it with the execution of adaptation and completion rules.

References

1. Garca, J.A.R.: jCOLIBRI: A multi-level platform for building and generating CBR systems. PhD thesis, Universidad Complutense de Madrid (Oct 2008)
2. DFKI: mycbr. `http://www.mycbr-project.net` (2011) (last verified 2011-06-10).
3. Attensity: Research & discovery - product home page. `http://www.attensity.biz/en/Applications-and-Services/Applications/Research-and-Discovery.html` (2011) (last verified 2011-06-10).
4. Hanft, A., Ihle, N., Newo, R.: Refinements for retrieval and adaptation of the CookIIS application. In Hinkelmann, K., Wache, H., eds.: Wissensmanagement. Volume 145 of LNI., GI (2009) 139–148
5. Hanft, A., Newo, R., Bach, K., Ihle, N., Althoff, K.D.: Cookiis a successful recipe advisor and menu creator. In Montani, S., Jain, L., eds.: Successful Case-based Reasoning Applications - I. Volume 305 of Studies in Computational Intelligence. Springer Berlin, Heidelberg (2010) 187–222
6. Bali, M.: Drools JBoss Rules 5.0 Developer's Guide. Packt., Birmingham (2009)
7. The Eclipse Foundation: SMILA– eclipse project. `http://www.eclipse.org/smila/` (2011) (last verified 2011-06-08).
8. The Eclipse Foundation: SMILA Wiki - Architecture Overview. `http://wiki.eclipse.org/SMILA/Architecture_Overview` (2011) (last verified 2011-06-08).
9. The Eclipse Foundation: Smila wiki - pipelets. `http://wiki.eclipse.org/SMILA/Documentation/Pipelets` (2011) (last verified 2011-06-09).
10. JBoss: Drools. `http://www.jboss.org/drools` (2011) (last verified 2011-06-10).
11. Schäfer, O.: Integration eines regelbasierten Systems über die OSGi Service Platform in SMILA. Bachelor thesis, Universität Hildesheim (2011)
12. Bogaerts, S., Leake, D.: IUCBRF: A framework for rapid and modular CBR system+development. Technical report, Indiana University, Bloomington (2005)
13. Minor, M., Bergmann, R., Görg, S., Walter, K.: Towards case-based adaptation of workflows. In Montani, S., Bichindaritz, I., eds.: ICCBR 2010. LNAI 6176, Springer-Verlag (2010) 421–435