A MDD approach for generating Rule-based Web Applications from OWL and SWRL

Joaquín Cañadas[1], José Palma[2] and Samuel Túnez[1]

[1] Dept. of Languages and Computation. University of Almeria. Spain
jjcanada@ual.es, stunez@ual.es
[2] Dept. of Information and Communications Engineering. University of Murcia. Spain
jtpalma@um.es

Abstract. Rule languages and inference engines incorporate reasoning capabilities to Web information systems. This paper presents a model-driven approach for the development of rule-based applications for the Web. The method is applied to ontology and rule specifications in OWL and SWRL, generating a rich, functional Web architecture based on the Model-View-Control architectural pattern and the JavaServer Faces technology, embedding a Jess rule engine for reasoning and inferencing new information. The proposal is described through an illustrative example. A tool supporting the ideas presented this paper has been developed.

Keywords: MDD, OWL, SWRL, Rule-based systems for the Web

1 Introduction

Rule-Based Systems (RBS) is the leading technology for the development of Knowledge-Based Systems from the beginnings of the Artificial Intelligence. RBS is a kind of software systems in which the human expert's knowledge applied for solving a complex task such as diagnosis, monitoring, assessment, and so on, is represented as a set of declarative production rules. Rules encode domain knowledge and business logic as condition-action pairs, whereas rule engines are able to interpret the rules and reason over the information loaded in the working memory, using some inference method to deduct new assertions and achieve a conclusion as the human expert would do [7,29].

Currently, a high interest exists in making the Web a more effective platform for intelligent systems. In this context, rule languages and inference engines provide Web information systems with reasoning capabilities [26]. Rules and ontologies play a key role in the architecture of the Semantic Web, as they are used to provide meaning and reasoning facilities to semantic Web applications [8].

Ontologies are typically used for domain modeling, in which a conceptualization of a particular domain is given. Although ontology formalisms are quite mature and the Web Ontology Language (OWL) [5] has become a standard, rule formalisms are an active area of research. The Semantic Web Rule Language (SWRL) [15] was proposed to enrich OWL providing a high-level abstract

syntax for rules in OWL. SWRL rules are of the form of an implication between an antecedent (body) and consequent (head). So when combining ontologies and rules for domain conceptualization and inference modeling, ontologies can describe the relevant concepts and relationships in an application domain and rules can be used to formalize the inference logic, increasing the amount of knowledge that can be represented in ontologies.

This paper presents a model-driven approach to develop rule-based Web applications based on OWL and SWRL specifications. The proposal applies Model-Driven Development (MDD) or Model-Driven Engineering (MDE) to automatically produce the implementation of a functional, rich Web architecture which embeds a rule engine for inferencing tasks. Although other formalisms for conceptual and rule modeling can be used in MDD of rule-based Web applications [3], the use of OWL and SWRL as specification formalisms in rule-based Web applications development becomes a reasonable choice since they can be considered as standard languages widely used by the community (mainly OWL), and there are many tools for editing and reasoning over OWL and SWRL specifications.

Recently, a tool to bridge the gap between OWL ontologies and MDE has been presented, the TwoUse Toolkit [3] [27], which has been developed using Eclipse Modeling Project[4]. It implements the Ontology Definition Metamodel (ODM) [24] and provides a graphical environment for specifying OWL and SWRL models. Since it is deployed as an Eclipse plugin, other Eclipse-based tools for MDD can be straightforward applied enabling us to design a model-driven process based on OWL and SWRL ontologies created with TwoUse.

In the proposed method, the functionality for the generated rule-based Web application is predefined to enable end-users to create, retrieve, update and delete instances (CRUD). In contrast to current tools for automatic generation of CRUD systems that perform those functions on relational databases, our contribution is that this functionality is executed on the rule engine working memory, enabling the execution of a forward-chaining inference mechanism to drive the reasoning process.

To put our proposal into practice, a supporting tool has been developed using MDD tools provided by the Eclipse Modeling Project. The resulting Web application, based on the Model-View-Controller (MVC) architectural pattern, embeds the Jess rule engine [12] to incorporate inference features. The proposed approach materializes *InSCo* [1], a methodology which intertwines knowledge engineering and software engineering approaches in hybrid intelligent information systems development.

The rest of this paper is organized as follows: Section 2 introduces the model-driven approach proposed and the supporting tool that implements it. Next, the mapping from OWL and SWRL to Jess is detailed in section 3. The architecture for the rule-based Web application generated is described in section 4. Some related work is reviewed in Section 5. Finally, the main conclusions and future work are summarized.

[3] http://code.google.com/p/twouse/
[4] http://www.eclipse.org/modeling/

2 Model-driven approach for rule-based Web applications

2.1 Process overview

The proposed approach uses OWL and SWRL ontologies created with TwoUse as source models for the model-driven process. We will focus on *OWL DL* sub-language of OWL. Classes, properties, and individuals define the structure and content of data. SWRL rules describe logical dependencies between the elements of the ontology refereed in the rule's antecedent and consequent.

Fig. 1 shows the proposed MDD schema for rule-based Web applications, which is divided into two processes. The first one (the bottom flow in Fig. 1) generates the implementation of the rule base in Jess, and the second one (the top flow in Fig. 1) produces the code for the Web architecture.

The development process starts with the specification of an OWL ontology of the problem domain enriched with SWRL rules, using the TwoUse editor. Such ontology is treated as a platform-independent model by the MDD process which produces two different results: (1) ontology and rules are transformed into a Jess rule base, a text file containing the set of templates, rules and facts in Jess syntax; (2) furthermore, a Web-based architecture is generated from the ontology and from an interaction and presentation model which is tightly coupled to the ontology. Web application code is based on the MVC architectural pattern, the JavaServer Faces (JSF) framework [13] and JBoss Richfaces components [16], and consist of a set of generated JavaBeans, JSF pages and configuration files.

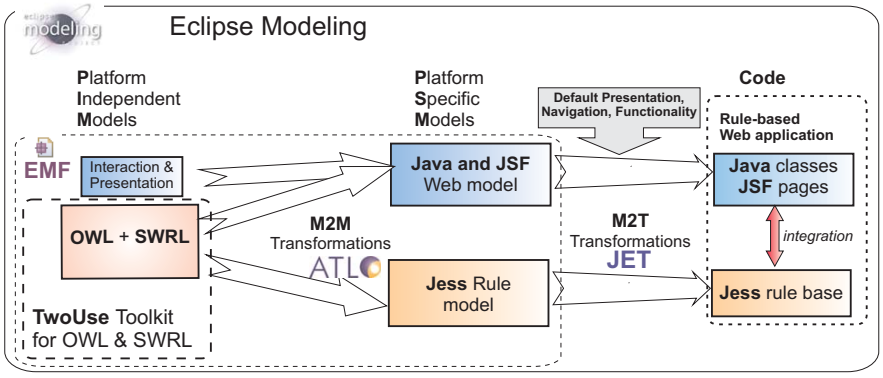

Fig. 1. MDD schema for rule-based Web system generation

Although the two MDD processes are executed independently, the final result must integrate the rule base into the Web application. This is done by generating the appropriate calls to the Jess API (Application Programming Interface) in the Java code obtained, entailing to embed the Jess rule engine into the Web application.

The generated Web application benefits of having the decision logic externalized from core application code, since uncoupling the rules from the source code increases scalability and maintainability of rule-based applications [10]. In addition, our development approach makes it possible for the two MDD processes to be executed separately, and therefore, any change in the rule model affecting only the rule logic (SWRL rules) without having an effect on the ontology (classes, properties, ...) can be translated to a new rule base without having to modify or regenerate anything else in the Web architecture. This approach makes Web applications easier to maintain and evolve.

2.2 Tool support

A tool supporting the proposed approach is being developed using Eclipse MDD tools and it is supported by the TwoUse toolkit for ontology edition. Using EMF[5] (Eclipse Modeling Framework) we have defined two metamodels for representing Jess and Java/JSF Web models, respectively.

Model-to-model (M2M) transformations are designed with ATL[6] (Atlas Transformation Language). The first M2M transformation (bottom flow in Fig. 1) maps an ontology and rule specification to a Jess platform-specific model. The second ATL transformation (top flow in Fig. 1) transforms the ontology and the interaction and presentation model into a Java/JSF Web specific model.

The outputs of both ATL transformations are the respective inputs of two model-to-text (M2T) transformations implemented in JET[7] (Java Emitter Templates). As a result, the code for the rule-based Web application is obtained. On one hand, source files with Jess rule base and facts, and on the other hand, the Web application components, the configuration files, the Java classes, and a Jess-Engine Bean which uses the Jess API to embed the rule engine into the architecture. Moreover, a set of JSP/JSF web pages are generated composing the user interface which is based on the RichFaces framework to add AJAX capability to JSF applications. Required Java libraries (Jess, JSF, Richfaces,...) and default configuration are injected to provide presentation templates and style sheets for pages, predefined navigation between them and default functionality for the generated Web application.

3 MDD of Jess rule base

This section describes the transformation from OWL and SWRL to Jess. Since OWL semantics is richer than semantics of production rule systems such as Jess, only a part of OWL can be represented in Jess. To illustrate the subset of OWL supported by the proposed transformation, the family relationships example [15] was used. Fig. 2 shows the family related classes created with the TwoUse OWL graphical editor.

[5] http://www.eclipse.org/modeling/emf/
[6] http://www.eclipse.org/m2m/atl/
[7] http://www.eclipse.org/modeling/m2t/?project=jet

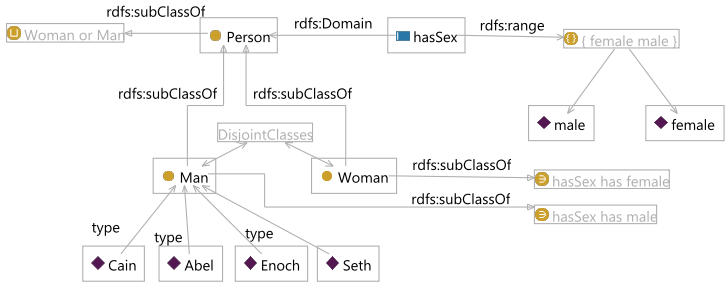

Fig. 2. Family classes in OWL

Domain concepts are represented as named classes, which can have subclasses. In the example the *Person* class has two subclasses, *Man* and *Woman*. Individuals in OWL are instances of classes declared as being a member of a specific class through the *type* link. *Cain*, *Abel*, *Enoch* and *Seth* are individuals of *Man* class. OWL provides several class extension constructs to define unnamed anonymous classes. The "oneOf" expression enables the definition of an enumerated class through the list of individuals that constitute the instances of the class. An example of enumerated class is {*female, male*}.

In OWL two types of properties are distinguished: datatype properties, relations between instances of classes and primitive data types (e.g. integer or string); and object properties, relations between instances of two classes. Each property has a domain and a range. In the family example, the *hasSex* object property has a domain of *Person* and a range of {*female, male*}, relating instances of the class *Person* to one of the possible enumerated instances. The default behavior in OWL indicate that a property can relate an instance of the domain to multiple instances of the range. To relate an individual to only one other individual a property is defined as *functional*, as example *hasSex*.

It is possible to further constrain the range of a property with property restrictions. The *"has-Value"* restriction specify classes based on the existence of particular property values. A property restriction can be treated as an anonymous OWL class, which means that it is possible to define another OWL class as a subclass of a property restriction. In the example, *Woman* is a subclass of *"hasSex has female"*. Similarly, *Man* is subclass of *"hasSex has male"*.

The family ontology defines several object properties to represent family relationships between individuals, as Fig. 3 shows. The *hasBrother* and *hasUncle* object properties have the domain of *Person* and the range of *Man*. *hasParent* has *Person* as domain and range. Those three properties are *non-functional*. Moreover, three property assertions are defined, *hasBrother(Cain, Abel)*, *hasBrother(Cain, Seth)* and *hasParent(Enoch, Cain)*.

The generation of Jess code from OWL is supported by a model-to-model transformation which maps elements of the OWL metamodel to elements of a Jess metamodel. Fig 4 (a) shows the Jess model obtained from the family

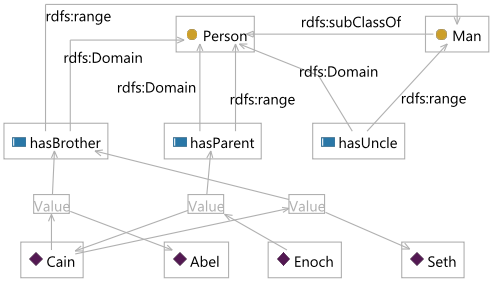

Fig. 3. Family object properties and assertions to individuals

ontology, focusing on the mapping from OWL classes and properties to Jess templates and slots. The execution of a model-to-text transformation to this Jess model generates the code listed on Fig 4 (b).

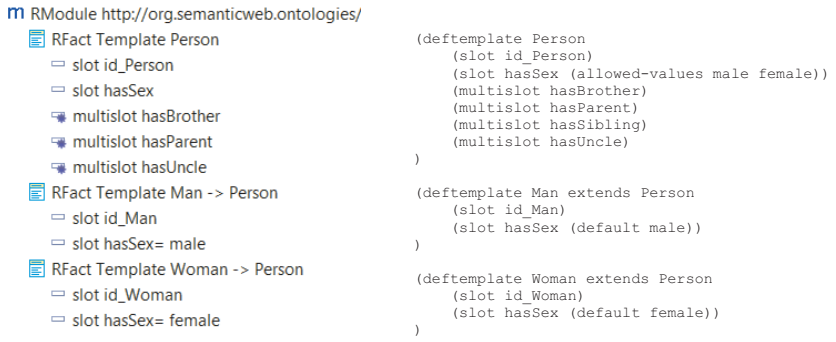

Fig. 4. a) Jess model of *templates* and b) final Jess code obtained

A summary of the ATL transformation rules defining that mapping are as follows. Every OWL named class is mapped to a Jess fact template. A subclass relation is represented through the *extends* template property, allowing only simple inheritance of templates. OWL properties, both datatype and object ones, are mapped to slots. Each slot is added to the template corresponding to the domain class of the property. Functional properties generate slots (single-valued) whereas non-functional properties generate multislots (multi-valued). When the range of a property is an enumerated defined through a "oneOf" OWL anonymous class, the enumeration literals obtained from the individuals linked to the enumeration are saved as a list of allowed-values for the slot. A *"has-Value"* restriction is transformed to a slot default value.

As Fig 4 shows, the style of the generated Jess code is quite different to the Jess code obtained in other tools like the Protege JessTab [9]. Instead of

generating a fact for every property in a propositional logic style, our proposal transforms properties to slots in templates, obtaining a result closer to an object oriented programming style. The reason for that design decision is that this style is more similar to the Java classes of the Web application that will embed the Jess rule engine, facilitating integration of the information managed in the different layers of the Web application (Jess, Java classes and Web pages).

Fig 5 (a) shows the Jess model obtained from the family ontology focusing on the mapping from individuals members of named classes to Jess facts definitions. All individuals are considered different of each other. Each individual is mapped to a fact definition. Each property assertion generates a value for a slot of the corresponding fact definition. In the example, the assertions *hasBrother(Cain, Abel)* and *hasBrother(Cain, Seth)* generate the *Cain* fact definition to have the *hasBrother* multislot with *Abel* and *Seth* values. The *deffacts* sentence grouping all fact definitions is loaded as the initial state of the Jess rule engine working memory.

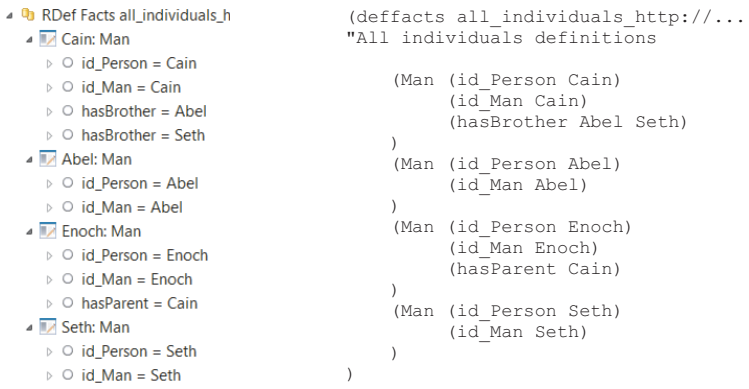

Fig. 5. a) Jess model of *facts* and b) final Jess code obtained

To finish the description of the use case, an example of SWRL rule is defined to infer the uncle relationship between family members. This rule is shown in Fig. 6 and it derives the value of the *hasUncle* property: for those individuals of Person type (*x1*) who has other individual of Person type (*x2*) related through the *hasParent* property, and that second Person (*x2*) has an individual of Man type (*x3*) related through the *hasBrother* property, the rule derives a new assertion staying the *hasUncle* relation between *x1* and *x3* individuals.

The rule antecedent is mapped to a set of pattern matchings and conditional expressions. In the rule consequent, each individual property atom maps to a *modify* function that update fact slots. The following code is generated from the SWRL uncle rule.

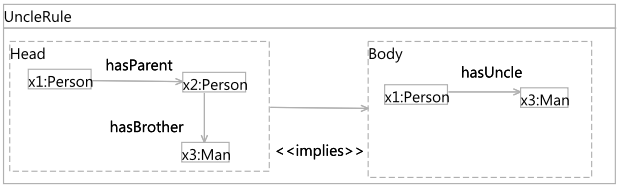

Fig. 6. SWRL rule defining *uncle* relationship in Family ontology

```
(defrule R1_UncleRule
  (declare (salience 10))
  ?x1<-(Person (id_Person ?x1_id_Person) (hasUncle $?x1_hasUncle)
               (hasParent  $?x1_hasParent))
  ?x2<-(Person (id_Person ?x2_id_Person) (hasBrother $?x2_hasBrother))
  ?x3<-(Man    (id_Man    ?x3_id_Man))
  (test (neq  ?x1_id_Person ?x2_id_Person ?x3_id_Man ))
  (test (not (member$ ?x3_id_Man ?x1_hasUncle)))
  (test (member$ ?x2_id_Person ?x1_hasParent))
  (test (member$ ?x3_id_Man ?x2_hasBrother))
 =>
  (modify ?x1 (hasUncle (create$ ?x1_hasUncle ?x3_id_Man)))
)
```

Each variable in the rule antecedent, *?x1*, *?x2* and *?x3*, is mapped to a pattern matching allowing the rule engine to match facts for the variables. The third and fourth test conditional expressions are obtained from the individual property atoms of the SWRL rule antecedent. Two more test conditional expressions are added, the first one checks that the three matched facts are different, and the second is necessary because without it the monotonic nature of the rule engine algorithm generates an infinite loop when the action has a modify function.

4 MDD of JSF Web architecture

A second MDD process is applied (see Fig. 1) to generate a Web architecture that embeds rules into a Web application. The generated application is divided in two layers: business and user interface. The business layer is composed of a set of Java classes that we can differentiate between the JavaBeans obtained from the named classes of the ontology, a set of helper classes implementing some necessary functionality, and a *JessEngineBean* class that implements the interaction with Jess through the Jess API, embedding the rule engine and the rule base into the Web application. The user interface layer is composed of a list of JSF web pages allowing end-users to interact with the system and accomplish the predefined functionality.

4.1 Architecture of rule-based Web applications

Fig. 7 shows the target architecture for the generated rule-based Web application.

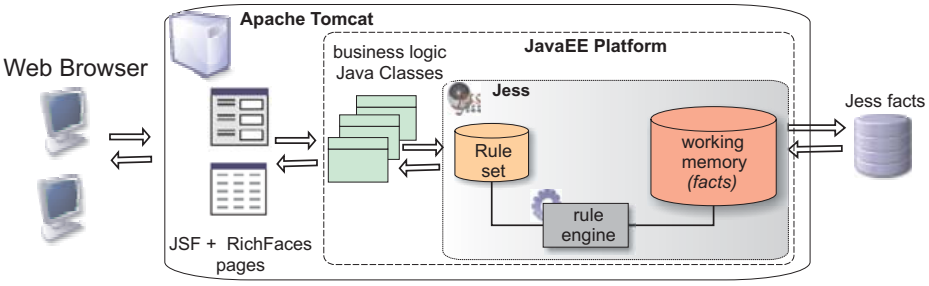

Fig. 7. Architecture of a rule-based Web application

The embedded rule engine manages the Jess rule base and a text file of persistent *facts*. Basically, the rule engine consists of three parts: the working memory which contains all the instances or *facts*, the rule set that includes the production rules, and the rule engine that checks what facts in the working memory match the rule conditions and executes the rules whose condition is matched as true. To make the facts persistent, they are stored on disk as a text file.

The Web application enables end-user to perform the most common functions for instance management: create, list, update and delete instances (CRUD). This functionality was predefined as a design decision of the proposed approach. The code generation process is designed to implement these functions since they constitute the basic information management tasks that any information system should offer. Currently there are many tools that automatically generate CRUD Web applications code from different sources, like from databases such as NetBeans[8], or from UML models such as MagicDraw M2Flex[9]. Those Web applications perform CRUD functions on relational databases. In contrast, our contribution provides a Web application where the information management functions are executed on the rule engine working memory, enabling the rule engine to run a forward-chaining inference mechanism that fires the rules which conditions are evaluated as true and executes rule's actions to deduct new information or modify existing one.

Fig. 8 shows a screenshot of the resulting Web application for the family example, in concrete the list of all instances of *Person* charged on the rule engine working memory. The list contains the four individuals defined in the OWL ontology and the three property assertions staying the brothers of *Cain* and the parent of *Enoch*. In addition, the *HasUncle* property has values for *Enoch* since the rule engine has derived those values firing the uncle rule. Using the Web application, end-users can add new persons, erase or modify persons using a form that shows all the properties of the person to edit. Some property values such as the *HasUncle*, are not editable by the user since it is the rule

[8] http://netbeans.org/
[9] http://www.model2code.com/

engine who obtains those values. This restriction is defined at design time using the interaction model, as described in next section.

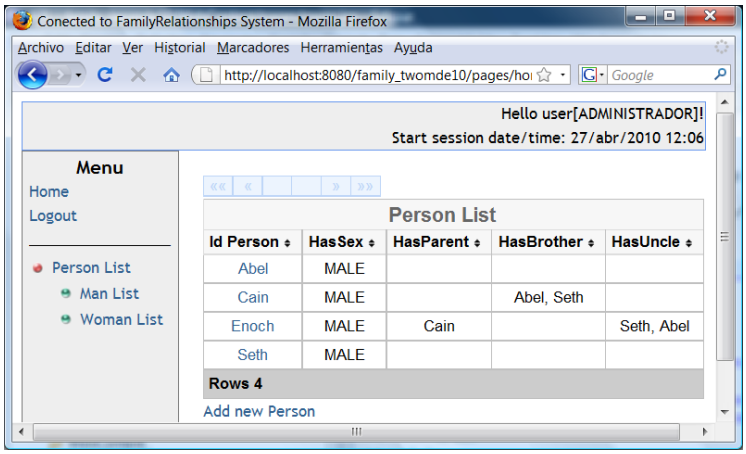

Fig. 8. Screenshot with the list of Persons

4.2 Generating the Web components

Fig 9 shows the metamodel proposed for defining Web models based on Java/JSF architecture. An ATL transformation maps OWL models to Web models in accordance with this metamodel. Basically the structure of the information managed by the Web application is defined from OWL classes, properties, and individuals. Each OWL named class is mapped to several elements: a JavaBean, a managed bean to be included in the configuration file, and two JSF pages, one for listing all instances of the class and other that shows all the individual's properties and their values enabling instance creation and edition.

Most of the user interface is predefined as well as navigation links between pages. However, some necessary features are included into an interaction and presentation model which is used by the ATL transformation to obtain the Web model. The edition of a specific individual implies that some properties are editable by the user, whereas other are derived by the rule engine inference process. For example, *hasBrother* property of an instance of *Person* is editable whereas the *hasUncle* value is not editable because it is obtained by a rule. This kind of restriction must be modeled in the interaction and presentation model that enables the specification of user interactivity and element's visibility features, enabling user interface customization. For example, it makes it possible to select what classes will appear in the application menu and what properties are included as columns of the table listing all individuals of that class (see Fig. 8).

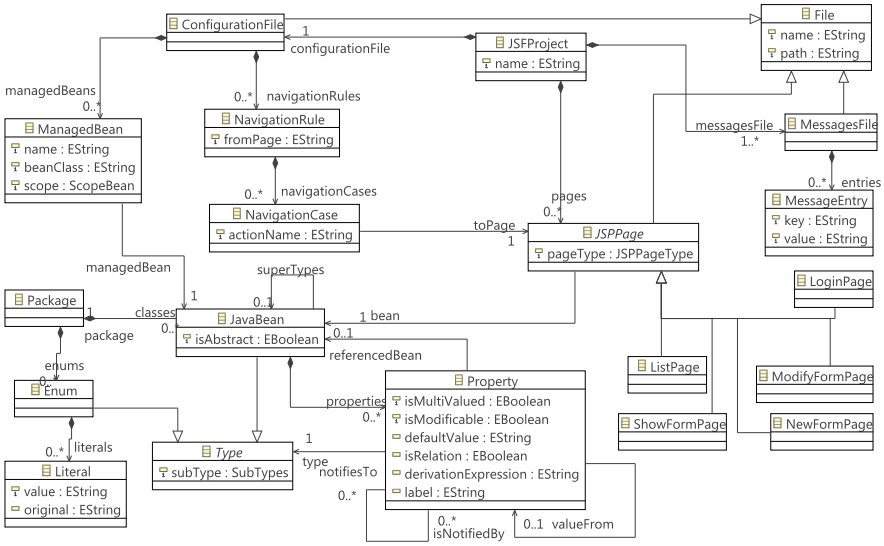

Fig. 9. Metamodel for the Java/JSF arquitecture

5 Related Work

Some previous work proposes the generation of Jess rules from ontology and rule models, such as OWL [20] and SWRL [25]. Those works focus on the definition of reasoners for OWL and Jess, where the main issue is to derive certain facts based on the OWL and SWRL semantics, respectively. The most important difference between those works and our proposal is that they run the generated rule base in a development tool such as the Protege JessTab [9], using a reasoner embedded in the environment to execute rules. In our work, we generate the Jess rule base and also the necessary code to embed the Jess rule engine into a functional Web application that can be used by end-users.

The definition of processes, techniques and models suitable for Web application development is the goal of Web Engineering. Methodologies such as UWE [17], WebML [11] and WebDSL [14], approach the design and development of Web applications providing tools for Web application development which are mainly based on conceptual models [4], focusing on content, navigation and presentation models as the most relevant concerns in the design of Web applications [21,19]. While Web engineering methodologies for traditional Web applications offer rather mature and established solutions, Semantic Web application methodologies are still in a development phase [2]. Several model-driven methodologies provide methods to generate Semantic Web Information Systems from specifications, such as SHDM [18], Hera [30], and WebML+Sem [2]. They identify different general design phases such as conceptual modeling, navigation modeling and presentation modeling, supporting in a high or low grade different semantic web technologies such as semantic model description, advanced query support,

flexible integration, ontology reasoning, and more. They also offer a wide support for ontology languages, basically all the models support both RDF and OWL. The main contribution in our proposal with respect to those methods is to give support to rule modeling in SWRL as a different aspect in modeling Web applications, defining the model-driven approach to produce Web applications embedding a rule engine. It enhances Web applications functionality providing the capability of inferring new information through the rule engine reasoning mechanism.

Currently the definition of a model-driven methodology for creating rule-based systems embedded in (semantic) Web applications is a working topic. In [6] the authors describe how to merge UML models and OWL ontologies for automatic business rules generation in Semantics of Business Vocabulary and Business Rules (SBVR) syntax [23]. Although they propose a general architecture for applying MDA and OWL ontologies to generate the rulesets in a target rule engine, at the moment of writing the paper they only generate the first abstraction version of business rules. Other work regarding MDD of Web applications integrating rules [28] describes MDD principles for rule-based Web services modeling using R2ML, and proposes an MDD approach for generating Web services from rule models. Whereas that proposal focuses on a Web services architecture, our work is based on a MVC architecture using the JSF framework.

6 Conclusion and future work

This paper applies MDD to rich Web system development, incorporating a Jess rule engine for inferencing and deriving new information. OWL and SWRL formalisms are used as modeling languages by the model-driven approach.

To illustrate the proposed method and supporting tool, a simple and well known use case has been used. The approach is planned to be evaluated in several domains such as pest control in agriculture and medical diagnosis.

Since expressiveness in rule-based production systems such as Jess is poorer than OWL and SWRL semantics, only a subset of those formalisms is supported. Issues related with the combination of rules and ontologies have also an effect on the proposed approach. For example, the semantics of OWL and SWRL adopts an open world assumption, while logic programming languages such as Jess are based on a closed world assumption. As a consequence, only DL-Safe SWRL rules [22] are transformed to Jess. DL-Safe SWRL rules are a restricted subset of SWRL rules that has the desirable property of decidability, by restricting rules to operate only on known individuals in an OWL ontology. Similarly, Jess rules only operate on facts in the working memory.

Other factor to take into account is the high time and effort necessary for developing a MDD supporting tool, which makes any model-driven approach an appropriate solution only when is useful to provide a tool that can be used in the construction of many similar systems, in many domains. In contrast, some strengths are obtained, such as the drastic cost reduction in time and effort for development rule-based Web applications since the tool makes all the hard work

for us. Also, an incremental development is favored with this approach since changes in the model can be directly reflected in the system implementation.

Future work extends the generated Web application with semantic Web functionalities, such as semantic search based on ontology classes hierarchy as well as instance search. To avoid the semantic gap between OWL and the rule engine, a solution would be the use of a reasoner such as Pellet[10] which directly interprets OWL and SWRL.

Acknowledgments. This work was supported by the Spanish Ministry of Education and Science under the project TIN2004-05694, and by the Junta de Andalucia (Andalusian Regional Govt.) project P06-TIC-02411.

References

1. del Águila, I.M., Cañadas, J., Palma, J., Túnez, S.: Towards a methodology for hybrid systems software development. In: Proceedings of the Int. Conference on Software Engineering and Knowledge Engineering (SEKE). pp. 188–193 (2006)
2. Brambilla, M., Facca, F.M.: Building semantic web portals with WebML. In: Baresi, L., Fraternali, P., Houben, G.J. (eds.) ICWE. Lecture Notes in Computer Science, vol. 4607, pp. 312–327. Springer (2007)
3. Cañadas, J., Palma, J., Túnez, S.: InSCo-Gen: A MDD Tool for Web Rule-Based Applications. In: Gaedke, M., Grossniklaus, M., Díaz, O. (eds.) ICWE. Lecture Notes in Computer Science, vol. 5648, pp. 523–526. Springer (2009)
4. Ceri, S., Fraternali, P., Matera, M.: Conceptual modeling of data-intensive web applications. Internet Computing, IEEE 6(4), 20–30 (2002)
5. Dean, M., Schreiber, G.: OWL Web Ontology Language Reference. W3C Recommendation (2004), available at http://www.w3.org/TR/owl-ref
6. Diouf, M., Maabout, S., Musumbu, K.: Merging model driven architecture and semantic web for business rules generation. In: Marchiori, M., Pan, J.Z., de Sainte Marie, C. (eds.) RR. Lecture Notes in Computer Science, vol. 4524, pp. 118–132. Springer (2007)
7. Durkin, J.: Expert Systems: Catalog of Applications. Akron (Ohio), Intelligent Computer Systems Inc. (1993)
8. Eiter, T., Ianni, G., Krennwallner, T., Polleres, A.: Rules and ontologies for the semantic web. In: Baroglio, C., Bonatti, P.A., Maluszynski, J., Marchiori, M., Polleres, A., Schaffert, S. (eds.) Reasoning Web. Lecture Notes in Computer Science, vol. 5224, pp. 1–53. Springer (2008)
9. Eriksson, H.: Using JessTab to integrate Protege and Jess. Intelligent Systems, IEEE 18(2), 43–50 (2003)
10. Frankel, D., Hayes, P., Kendall, E., McGuinness, D.: The Model Driven Semantic Web. In: 1st International Workshop on the Model-Driven Semantic Web (MDSW2004), Monterey, California, USA (2004)
11. Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. Morgan Kaufmann, 1 edn. (Dec 2002)
12. Friedman-Hill, E.: Jess in Action: Java Rule-Based Systems. Manning Publications (2003)

[10] http://clarkparsia.com/pellet

13. Geary, D., Horstmann, C.S.: Core JavaServer Faces. Prentice Hall, 2 edn. (2007)
14. Groenewegen, D.M., Hemel, Z., Kats, L.C.L., Visser, E.: WebDSL: a domain-specific language for dynamic web applications. In: Harris, G.E. (ed.) OOPSLA Companion. pp. 779–780. ACM (2008)
15. Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language combining OWL and RuleML. W3C. Available at http://www.w3.org/Submission/SWRL/ (2004)
16. JBoss: RichFaces (2007), http://www.jboss.org/jbossrichfaces/
17. Koch, N., Knapp, A., Zhang, G., Baumeister, H.: UML-based web engineering. an approach based on standards. In: Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (eds.) Web Engineering: Modelling and Implementing Web Applications, pp. 157–191. Human-Computer Interaction Series, Springer London (2008)
18. Lima, F., Schwabe, D.: Application modeling for the semantic web. In: LA-WEB. pp. 93–102. IEEE Computer Society (2003)
19. Linaje, M., Preciado, J.C., Sanchez-Figueroa, F.: Engineering rich internet application user interfaces over legacy web models. Internet Computing, IEEE 11(6), 53–59 (2007)
20. Mei, J., Bontas, E.P., Lin, Z.: OWL2Jess: A transformational implementation of the OWL semantics. In: Chen, G., Pan, Y., Guo, M., Lu, J. (eds.) ISPA Workshops. Lecture Notes in Computer Science, vol. 3759, pp. 599–608. Springer (2005)
21. Moreno, N., Meliá, S., Koch, N., Vallecillo, A.: Addressing new concerns in model-driven web engineering approaches. In: Bailey, J., Maier, D., Schewe, K.D., Thalheim, B., Wang, X.S. (eds.) WISE. Lecture Notes in Computer Science, vol. 5175, pp. 426–442. Springer (2008)
22. Motik, B., Sattler, U., Studer, R.: Query Answering for OWL-DL with rules. Journal of Web Semantics 3(1), 41–60 (2005)
23. Object Management Group: Semantics of Business Vocabulary and Business Rules (SBVR). http://www.omg.org/spec/SBVR/1.0 (2008)
24. Object Management Group: Ontology Definition Metamodel. Version 1.0. http://www.omg.org/spec/ODM/1.0/ (2009)
25. O'Connor, M.J., Knublauch, H., Tu, S.W., Grosof, B.N., Dean, M., Grosso, W.E., Musen, M.A.: Supporting rule system interoperability on the semantic web with swrl. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) International Semantic Web Conference. Lecture Notes in Computer Science, vol. 3729, pp. 974–986. Springer (2005)
26. Paptaxiarhis, V., Tsetsos, V., Karali, I., Stamotopoulos, P.: Developing Rule-Based web applications: Methodologies and tools. In: Handbook of Research on Emerging Rule-Based Languages and Technologies: Open Solutions and Approaches, pp. 371–392. IGI Global (2009)
27. Parreiras, F.S., Staab, S., Winter, A.: TwoUse: integrating UML models and OWL ontologies. Tech. rep., Universität Koblenz-Landau (2007)
28. Ribarić, M., Gašević, D., Milanović, M., Giurca, A., Lukichev, S., Wagner, G.: Model-driven engineering of rules for web services. In: Lämmel, R., Visser, J., Saraiva, J. (eds.) GTTSE. Lecture Notes in Computer Science, vol. 5235, pp. 377–395. Springer (2007)
29. Russell, S.J., Norvig, P.: Artificial intelligence: a modern approach. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1995)
30. Vdovjak, R., Frasincar, F., Houben, G.J., Barna, P.: Engineering Semantic Web Information Systems in Hera. J. Web Eng. 2(1-2), 3–26 (2003)