# A Model-Driven Method for automatic generation of Rule-based Web Applications

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 Communication Engineering. University of Murcia. Spain
jtpalma@um.es

**Abstract.** Rule languages and inference engines incorporate reasoning capabilities in Web information systems. In this paper, a Model-Driven Development (MDD) approach for automatic code generation of rule-based Web applications is proposed. A rule-based model specifying domain expert knowledge and business logic through production rules (if-condition-then-action) becomes the source model for the development approach. Demonstrating our proposal, a tool supports the creation of rule models and the automatic execution of model-to-model and model-to-code transformations. As a result, a rich, functional, rule-based Web architecture is generated, based on the Model-View-Controller architectural pattern and the JavaServer Faces technology, and integrating a Jess rule engine to perform inference tasks.

**Key words:** Model-Driven Development, Web Applications, Rule-based systems

## 1 Introduction

The design of rule languages and inference engines to provide Web information systems with reasoning capabilities is an important Semantic Web research topic [1] in which production rules (if-condition-then-action) play a leading role, since they enable a declarative representation of domain expert knowledge and business logic. Rule engines deal with rule bases and execute inference methods for firing the right rules in order to deduce information and conclude new results [2].

This paper addresses the development of rule-based systems embedded in Web applications to provide Web systems with inference capabilities, applying a model-driven approach to automate the development process of rule-based Web applications.

The terms Model-Driven Architecture (MDA) [3] and Model-Driven Development (MDD) [4] refer an approach of software development that uses models as first class entities, enabling the definition and automatic execution of transformations between models and from models to code. The creation of metamodels for specifying modeling languages is a basic task in MDA/MDD. Also the

specification of transformations between models, called model-to-model (M2M) transformations, and from model to code, called model-to-text (M2T) transformations. The main advantage of this approach of software development is that MDD tools enable these transformations to be specified and executed automatically, using supporting languages and tools for MDA/MDD. This development approach is currently being applied to many domains in software development, such as embedded systems, Web engineering, Ontology Engineering, and more. However it has some limitations because it is relatively new, supporting tools for MDD are not mature enough, and it introduces some rigidity since writing models is not as flexible and expressive as writing source code.

In this work Conceptual Modeling Language (CML) [5] is used as rule modeling formalism, a language for knowledge representation defined by the CommonKADS methodology [6]. It enables the specification of the domain ontology and a set of production rules which are bound to ontology concepts. Models written in this formalism are independent of any implementation technology, and therefore, can be used as the source model in a model-driven approach.

To put our proposal into practice, a supporting tool developed using several tools provided by the Eclipse Modeling Project[3] applies a model-driven approach to rule models and automatically generates the implementation of a functional rule-based Web application. The resulting Web architecture is based on the Model-View-Controller (MVC) architectural pattern and the JavaServer Faces (JSF) framework [7], and incorporates rich JBoss Richfaces components [8] to enhance the user interface with AJAX (Asynchronous JavaScript And XML) capabilities. The Jess rule engine [9] is embedded in the Web architecture to provide inference features. The functionality of the rule-based Web application is predefined to create, read, update and delete instances (CRUD). In contrast to current tools for automatic generation of CRUD systems that perform those functions on relational databases, the contribution of our approach is that CRUD operations are executed on the Jess rule engine working memory, enabling the inference mechanism to execute a forward-chaining inference mechanism to drive the reasoning process.

The proposed approach materializes *InSCo* [10], a methodology which intertwines knowledge engineering and software engineering approaches in hybrid intelligent information systems development.

This paper is organized as follows: Section 2 introduces rule-based systems and rule modeling languages for the Web. Next, the rule-based modeling approach for specifying the Web applications proposed is described in section 3. After that, the model-driven method for rule-based Web application development is detailed in Section 4. The MDD support tool is presented in Section 5. Section 6 describes related work, and finally main conclusions and future work are summarized.

[3] http://www.eclipse.org/modeling/

## 2 Overview of Rule-based systems and rule modeling

Rule-based systems originated in Artificial Intelligence, as the kind of expert or knowledge-based system that use rules as knowledge representation formalism. In this kind of software system, 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. Rule engines are able to interpret the rules, and reason using some inference method to come to a conclusion as the human expert would do [11, 12].

In general, a rule-based system consists of a set of production rules, a working memory and an inference engine. The rules encode domain knowledge and business logic as condition-action pairs. The working memory initially represents the system input, but the actions that occur when rules are fired can cause the state of the working memory to change. The inference engine runs a reasoning method to fire rules, typically forward and backward chaining mechanisms. The execution of the action part of a rule involves inferring new data.

More recently, the software engineering community has also focused on rules as a proper formalism for representing business logic in software systems. Today these two points of view have merged, favoring the widespread adoption of rule-based systems and business rules in the implementation of complex decision-making processes [13].

Rule formalisms are an active area of research addressing the development rule languages and inference engines to add reasoning to complex information systems. The Object Management Group (OMG) proposed the Ontology Definition MetaModel [14] and Production Rule Representation [15] as standard metamodels for introducing both technologies in the context of MDA/MDD. Relevant initiatives to standardize and exchange rules are the Rule Markup Initiative (RuleML) [16], the Semantic Web Rule Language (SWRL) [17], the REWERSE Rule Markup Language (R2ML) [18], and the Rule Interchange Format (RIF) [19].

We use CML as the rule-modeling language because, although it is currently not one of the most common options for rule modeling, it has several features desirable for production-rule formalisms. It enables unified representation of ontologies and rules, in which rules and ontology are naturally related. It meets the requirements of rule representation formalisms, such as modeling rule antecedent, rule consequent, named rules, and rulesets, binding rules to ontology concepts, and so on. And finally, it is simpler and easier to use than other formalisms, although this may mean less expressiveness in certain situations.

## 3 Modeling Rule-based Web applications

The proposed model-driven approach for rule-based Web application development focuses on introducing rule modeling in the specification of Web applications. However, other modeling concerns related to Web design features must be also considered, powering the automatic code generation process. The CML

model describing the ontology and rule model is presented at a conceptual level, whereas interaction and presentation features are specified at a Web design level.

### 3.1 Conceptual rule-based modeling

The CML formalism for knowledge modeling entails the specification of simplified domain ontologies and production rules. A CML (domain knowledge) model is basically composed of two elements, domain schemas and knowledge bases. Domain concepts, binary relationships, rule types and value types (enumerated literals) are modeled in a domain schema. A knowledge base is composed of instances of concepts, instances of relationships called tuples, and instances of rules. Figure 1 shows the domain knowledge model components.

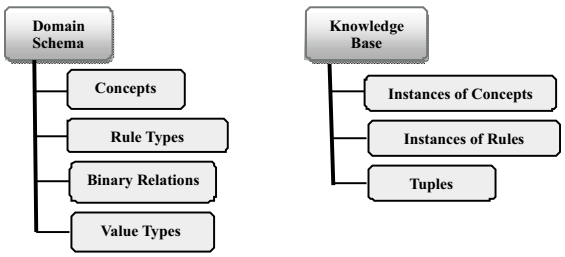


**Fig. 1.** Domain Knowledge structure

CML was originally defined as a textual notation by means of an abstract grammar described in EBNF (Extended Backus-Naur Form). To use CML in the context of MDD, we have specified a metamodel for CML. The main difference between this formalism and other conceptual modeling approaches in software engineering, such as UML class diagrams, is its ability to model production rules with rule type and the rule instance constructors. A rule type describes the structure of a set of rules through the specification of the ontology types bound to the rule antecedent and consequent. Rule types are particularized into rule instances which represent specific, logical dependencies between rule antecedent and consequent concept attributes.

### 3.2 Web design modeling

CML models are enriched with interaction and presentation characteristics to specify rule-based Web applications design features.

Interaction features enable the specification of user interactivity through a set of properties associated to CML constructors. The following properties dealing with attribute management will illustrate some interaction characteristics:

– *isDerived*. This property is set to *true* when the attribute value is inferred by the rule engine, so it cannot be edited by the user.

– *notifiesTo* and *isNotifiedBy*: These properties are used to indicate what attributes must be refreshed by the re-rendered AJAX facility in a user event, for example a mouse click or a change in the attribute value.

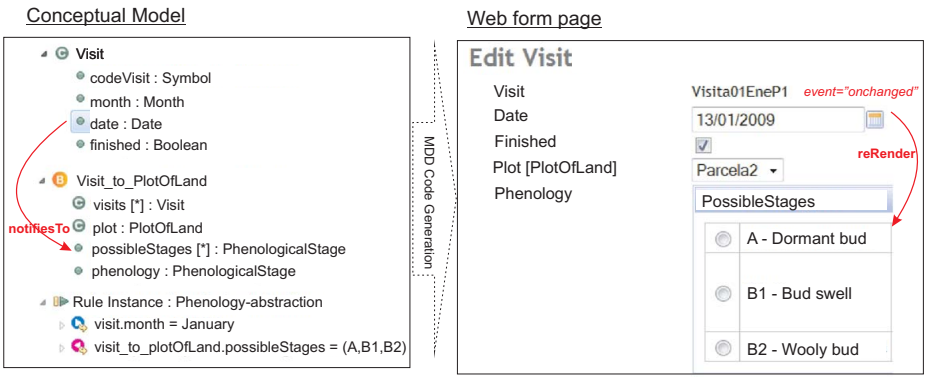


**Fig. 2.** Modeling interaction with *NotifiesTo* property

Figure 2 shows an example of how the interaction between two attributes defined in the conceptual model is specified using the *isDerived* and *notifiesTo* properties, showing how they affect Web forms for editing instances. The example is taken from SAVIA, a decision-support system for pest control in greenhouse crops and grapes that is being developed by applying rule-based modeling and the proposed model-driven approach for rule-based Web system development. The left side of Figure 2 shows a selection of SAVIA conceptual rule model elements. In particular, a concept called *Visit*, a relationship called *Visit_to_PlotOfLand*, and an example of rule instance belonging to the group of rules that specify the possible phenological stages of the crop depending on the date of the visit. The concrete rule instance is: *"if the month of the Visit is January then the possibleStages are (A, B1, B2)"*. Focusing on interaction specification, the *possibleStages* attribute is derived since it is inferred by the rule engine when it fires rules such as the one above. And the attribute *date* of visit notifies *possibleStages*, making that when the event *onChange* happens in the Web form date field, then an action makes the rule engine run and the list of *possibleStages* is re-rendered, updating the list with the new values determined by the rule engine.

Presentation features specify the conceptual model element's visibility properties, enabling user interface customization. For example, this makes it possible to select what concepts will appear in the application menu, and what attributes are included as columns of tables showing all instances of a concept type.

# 4 MDD for Rule-based Web applications

## 4.1 General perspective

Figure 3 shows the proposed MDD schema for rule-based Web applications, which is divided into two processes. The first one (the bottom flow in Fig. 3) generates the implementation of the rule base in a rule implementation technology, and the second one (the top flow in Figure 3) produces the code for the Web architecture.

The development process starts with the specification of a conceptual rule model which defines the domain ontology and the set of rules using an platform-independent formalism such as CML. Application of the model-driven approach produces two different results. One one hand, ontology and rules are transformed into Jess, which supports the development and deployment of rule-based systems tightly coupled to Java applications. As a result, a Jess rule base, a text file containing the set of rules converted to Jess syntax, is generated.

Furthermore, a Web-based architecture is generated from the CML model extended with the interaction and presentation features. Web application code is based on the MVC architectural pattern and the JavaServer Faces (JSF) framework, producing a set of JavaBeans classes and JSP (Java Server Pages).

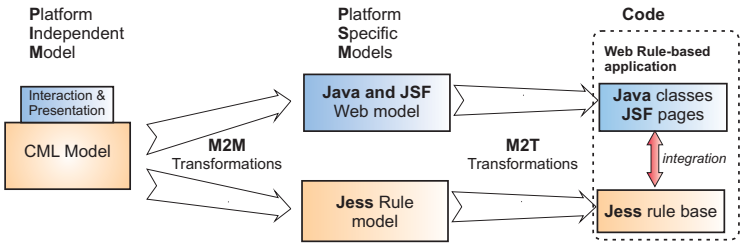


**Fig. 3.** MDD schema for Rule-based Web system generation

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

The rule-based Web application generated 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 [20]. Our approach makes it possible for the two MDD processes to be executed separately, and therefore, any change in the rule model affecting only to rule logic (rule instances) but without affecting to the structure of information (concepts, relationships, and so on) 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.

### 4.2 MDD of Jess rules

The first transformation of Jess rules in MDD involves the CML source model being translated into a platform-specific model based on a Jess rules metamodel, using an M2M transformation. The metamodel proposed for Jess rules (Figure 4) is an extended version of a simple rule metamodel for rule-based systems described in [21].

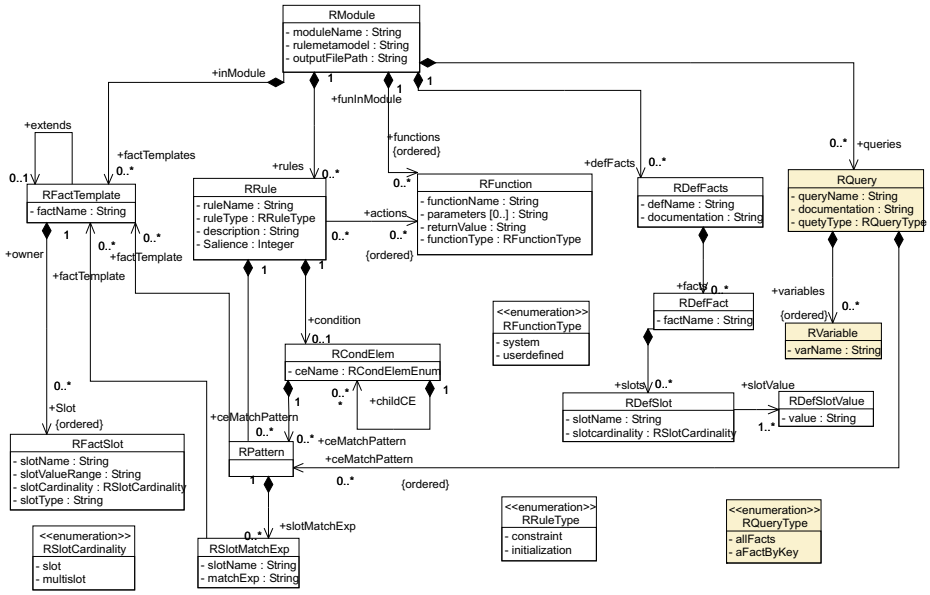


**Fig. 4.** Jess Rules Metamodel

In the Jess rules metamodel, a metaclass is defined for each Jess language element. The root element of a Jess rule model is *RModule*. A module contains fact templates, rules, functions, facts and queries. The *RFactTemplate* metaclass models fact templates, the Jess constructor for storing information.

*RRule* enables the representation of rules. A rule has a *ruleName*, a property called *salience* that determines the order in which applicable rules are fired by the rule engine, a containment reference *condition* representing the rule's condition part (antecedent), and a reference called *actions* representing the rule's action part (consequent). *RPattern* and *RSlotMatchExp* metaclasses define pattern matching expressions in rule conditions. Actions are function calls that assert new facts, or retract or modify existing facts.

Facts are defined by the *RDefFacts* metaclass. Facts are acquired from instances of concepts in the CML source model. Finally, *RQuery* models queries to consult the working memory at runtime.

The mapping from CML rule-based models to Jess rule models is designed by a M2M transformation which maps each CML metamodel constructor to one or several Jess Rule metamodel elements. The Jess rule model generated by the M2M transformation is the source model for a M2T transformation which automatically generates the Jess rule base source code, producing a Jess file (.clp) with a code for every element included in the Jess rule model. The M2T transformation is designed using JET, as described later in this paper.

### 4.3 MDD of JSF Web architecture

A second MDD process is applied (see Figure 3) to generate a Web architecture that integrates rules into a Web application. In this process, Jess rules can be integrated into the Web application, since both the Jess rule base and the Web architecture are generated from the same CML model.

Figure 5 shows the proposed target architecture for rule-based Web applications, based on the MVC architecture pattern, the JSF framework and rich AJAX JBoss Richfaces components.

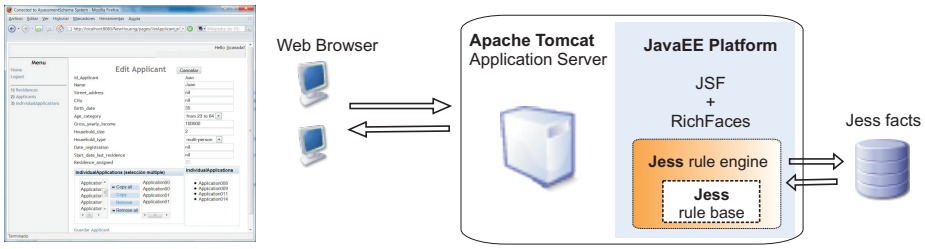


**Fig. 5.** Rule-based Web application architecture

The integrated rule engine manages the Jess rule base and the text file containing persistent *facts*. The Web application enables the user to perform four basic predetermined functions, create new instances, read the current list of instances, update and delete instances. That CRUD operations are executed on the Jess rule engine working memory, enabling the inference mechanism to fire appropriate rules when necessary. The rule engine executes a forward-chaining inference mechanism to drive the reasoning process, firing the rules with conditions evaluated as true, and executing their actions to infer new information or modify existing one.

A metamodel for the JSF Web architecture was designed. In the M2M and M2T transformations from a CML model to a JSF model and finally to code, each concept is mapped to several elements, a JavaBean class, a JSF page for instance creation and edition, a JSF page for listing all instances of that concept type, and a managed bean to be included in the configuration file. Interaction and presentation features are taken into account at this level in model-driven processes.

As a result, the use of both rules and AJAX technology improves the creation and edition of instances in the Web application. Since Web forms are implemented with AJAX RichFaces components, each single form value can be validated and submitted individually as it is entered. This facility entails the rule engine firing suitable rules and inferring new information that drives the instance creation or edition, for example, updating choice-field values.

## 5 Tool Support: *InSCo-Gen*

Our rule-based Web application model-driven development approach is demonstrated by our proof-of-the-concept, the *InSCo-Gen* tool. *InSCo-Gen* was developed using MDD tools provided by the Eclipse Modeling Project. Models and metamodels were defined using the Eclipse Modeling Framework (EMF[4]), including three metamodels, the CML metamodel for conceptual models, the Jess Rule metamodel used for representing Jess platform-specific models, and the JSF metamodel used by Web-based specific models.

Conceptual models conforming to the CML metamodel are created using the built-in reflective EMF editor. In order to improve model specification, the reflective editor is customized using Exeed (EXtended Emf EDitor) [22], a plugin which can modify editor default icons and labels, adding Exeed annotations to the metamodel. A screenshot with a model created with this editor is shown in Figure 2.

Modeling certain aspects of Web design, such as interaction and presentation, is implemented in different ways. Whereas interaction features are added to the conceptual CML metamodel through metaclass properties, presentation is defined by a set of XML configuration files, which can be edited by the developer before generating the Web application code.

Two M2M transformations are designed with Atlas Transformation Language (ATL[5]). The first one maps a CML model to a Jess platform-specific model. The second one transforms a CML model into a JSF-specific model.

The outputs of both ATL transformations are the inputs of two M2T transformations implemented with Java Emitter Templates (JET[6]). As a result, *InSCo-Gen* automatically produces the Web application code, on one hand, source text files with Jess rules and facts, and on the other, the Web application components, the faces-config.xml and web.xml configuration files, the Java Beans for model classes, and a Jess-Engine Bean which uses the Jess Java API (Application Programming Interface) to integrate the rule engine into the architecture. Moreover, a set of JSP/JSF web pages are generated for the user interface. These pages are based on the RichFaces library [8], an open source framework that adds AJAX capability to JSF applications.

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

## 6 Related Work

Our proposal uses the CML rule and ontology modeling formalism as the MDD source model. To put CML into the MDD framework, we defined a metamodel for CML. The definition of a UML Profile for the specification of CML knowledge models is addressed in [23] where the authors also discusses the possible mapping of the profile elements to a Jess platform specific model.

Some previous work has proposed the generation of Jess rules from ontology and rule models, such as OWL (Ontology Web Language) [24] and SWRL (Semantic Web Rule Language) [25]. These proposals focus on Jess code generation without applying a genuine MDD/MDA approach. But the most important difference between the proposal presented in this paper and those publications is that they do not integrate Jess into a functional Web application, so the Jess rule base generated must be run in a development tool using a shell, such as the Protege JessTab [26].

An MDD approach to Web Applications based on MVC and JavaServer Faces is described in [27]. Existing Web Engineering methods, such as UWE [28], WebML [29] and WebDSL [30], approach the design and development of Web applications addressing such concerns as structure, presentation, navigation. However, they do not consider rule modeling in Web application development. Our proposal focus on introducing rule modeling in this context, and we do not consider other concerns of Web application modeling such as navigation model, since we simplify the functionality to CRUD operations and, therefore, types and navigation links are fixed and preset.

Regarding MDD of Web applications integrating rules, [31] describes MDD principles for rule-based Web services modeling using R2ML, and proposes an MDD approach for generating Web services from rule models. Whereas this proposal focuses on a Web services architecture, our work is based on a MVC architecture using the JSF framework.

## 7 Conclusions and future work

In this paper, rule-based and model-driven techniques are intertwined for the development of rule-based Web applications. The main contribution of our work is to enrich the specification of Web applications with a rule modeling formalism, introducing a new concern in Model-Driven Web Engineering. A model-driven approach for generating Web implementation enhanced with inference features is described and demonstrated by an MDD tool.

The resulting rule-based Web architecture implements the MVC architectural pattern using the JavaServer Faces framework, and incorporates rich JBoss Richfaces components to enhance the user interface with AJAX capabilities. The Jess rule engine is embedded in the Web application to provide inference capabilities. Our proposal does not include a navigation model, since application functionality is predetermined by CRUD functions.

Due to the declarative nature of rules, the decision logic is externalized from core application code producing Web applications easier to maintain and evolve.

The approach is being evaluated through its use in the development of a Web decision-support system for pest control in agriculture, which makes recommendations to growers and technicians about the necessity of treating a specific pest or disease in grapes.

As future work, it is planned to use other ontology and rule modeling languages such as OWL and SWRL as source models for the model-driven approach, and define interoperability modules with other rule formalisms. Different rule platforms, such as JBoss Rules [32], will be also considered as a target rule technology. The Web application generated, which is aimed at enriching the architecture with database facilities, will be improved to provide a complete persistence layer.

**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. 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. Volume 5224 of Lecture Notes in Computer Science., Springer (2008) 1–53
2. Brachman, R.J., Levesque, H.J.: Knowledge representation and reasoning. Morgan Kaufmann, San Francisco (2004)
3. Object Management Group: MDA Guide Version 1.0.1. OMG document: omg/2003-06-01 (2003)
4. Mellor, S., Clark, A., Futagami, T.: Model-Driven Development - Guest editors introduction. IEEE Software **20**(5) (Sep-Oct 2003) 14–18
5. Anjewierden, A.: CML2. Technical Report 11, University of Amsterdam (1997) URL: http://www.swi.psy.uva.nl/projects/kads22/#cml2.
6. Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., de Velde, W.V., Wielinga, B.: Knowledge Engineering and Management: The CommonKADS Methodology. The MIT Press, Cambridge (2000)
7. Sun Microsystems: JavaServer Faces http://java.sun.com/javaee/javaserverfaces/.
8. JBoss: RichFaces (2007) http://www.jboss.org/jbossrichfaces/.
9. Friedman-Hill, E.: Jess in Action: Java Rule-Based Systems. Manning Publications (2003)
10. 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). (2006) 188–193
11. Durkin, J.: Expert Systems: Catalog of Applications. Akron (Ohio), Intelligent Computer Systems Inc. (1993)
12. Russell, S.J., Norvig, P.: Artificial intelligence: a modern approach. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1995)
13. Object Management Group: Semantics of Business Vocabulary and Business Rules (SBVR). http://www.omg.org/spec/SBVR/1.0 (2008)
14. Object Management Group: Ontology Definition Metamodel RFP (2003) Available: http://www.omg.org/cgi-bin/doc?ad/2003-03-40.

15. Object Management Group: Production Rule Representation RFP (2003) Available: http://www.omg.org/cgi-bin/doc?br/2003-09-03.
16. RuleML: The Rule Markup Initiative (2001) URL: http://www.ruleml.org.
17. 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 www.w3.org/Submission/2004/SUBM-SWRL-20040521 (2004)
18. REWERSE Working Group I1: R2ML -The REWERSE I1 Rule Markup Language (2006) URL: http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=R2ML.
19. Rule Interchange Format Working Group: RIF Use Cases and Requirements. W3C Working Draft (2006) URL: http://www.w3.org/TR/rif-ucr/.
20. 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)
21. Chaur G. Wu: Modeling Rule-Based Systems with EMF. Eclipse Corner Articles http://www.eclipse.org/articles/ (2004)
22. Kolovos, D.S.: Exeed: EXtended Emf EDitor - User Manual. http://www.eclipse.org/gmt/epsilon/doc/Exeed.pdf (2007)
23. Abdullah, M., Benest, I., Paige, R., Kimble, C.: Using unified modeling language for conceptual modelling of Knowledge-Based systems. In: Conceptual Modeling - ER 2007. (2007) 438–453
24. Mei, J., Bontas, E.P., Lin, Z.: OWL2Jess: A Transformational Implementation of the OWL Semantics. Lecture Notes in Computer Science **3759** (2005) 599–608
25. OConnor, M., Knublauch, H., Tu, S., Grosof, B., Dean, M., Grosso, W., Musen, M.: Supporting Rule System Interoperability on the Semantic Web with SWRL. Lecture Notes in Computer Science **3759** (2005) 974–986
26. Eriksson, H.: Using JessTab to integrate Protege and Jess. Intelligent Systems, IEEE **18**(2) (2003) 43–50
27. Distante, D., Pedone, P., Rossi, G., Canfora, G.: Model-Driven development of web applications with UWA, MVC and JavaServer faces. In: L. Baresi, P. Fraternali, and G.-J. Houben (Eds.): ICWE 2007, LNCS. Volume 4607., Springer, Heidelberg (2007) 457–472
28. Koch, N., Knapp, A., Zhang, G., Baumeister, H.: Uml_based web engineering: An approach based on standards. In: Web Engineering: Modelling and Implementing Web applications. Human-Computer Interaction Series. Springer, Berlin (dec 2007)
29. Fraternali, P., Bongio, A., Brambilla, M., Comai, S., Matera, M.: Designing Data-Intensive Web Applications. 1 edn. Morgan Kaufmann (December 2002)
30. Groenewegen, D.M., Hemel, Z., Kats, L.C., Visser, E.: WebDSL: a domain-specific language for dynamic web applications. In: Companion to the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications, Nashville, TN, USA, ACM (2008) 779–780
31. Ribarić, M., Gašević, D., Milanović, M., Giurca, A., Lukichev, S., Wagner, G.: Model-Driven engineering of rules for web services. In: Generative and Transformational Techniques in Software Engineering II: International Summer School, GTTSE 2007, Braga, Portugal, July 2-7, 2007. Revised Papers, Springer-Verlag (2008) 377–395
32. JBoss: Drools documentation http://www.jboss.org/drools/documentation.html.