UML-Based Rule Modeling with Fujaba

Sergey Lukichev[1], Gerd Wagner[1]
[1]Institute of Informatics, Brandenburg University of Technology at Cottbus, Germany
Lukichev@tu-cottbus.de, G.Wagner@tu-cottbus.de

ABSTRACT

In this paper we describe visual rule modeling tool Strelka, which is implemented as a Fujaba plugin. The modeling tool supports a UML-Based Rule Modeling Language (URML). It extends standard UML metamodel with a concept of a rule. We discuss an issue of a UML-based rule modeling, present rule metamodel, describe implementation of a Fujaba plugin and give examples of business rules, modeled using Strelka.

1. INTRODUCTION

Rules are becoming increasingly important in business modeling and requirements engineering, and as a high level programming paradigm. They are widely recognized to play an important role in the Semantic Web and are a critical technology component for the early adoption and applications of knowledge-based techniques in e-business, especially enterprize integration and B2B e-commerce. A lot of work has been conducted in the area of visual representation of business vocabularies and the mainstream technology is MOF/UML, which allows visualization of domain concepts by means of, for instance, UML class diagrams. In the area of emerging rule technologies for the Semantic Web and for Business Rules there are relatively few approaches and tools for visual rule modeling. We argue, that rule modeling language should allow visual rule expressions, which can be understood by domain experts or by existing software engineers without extensive technical training. We base our modeling approach on UML since it is widely adopted modeling language. Since rules are built on facts, and facts are built on concepts as expressed by terms[1], a UML-based rule modeling approach is a natural extension of UML. To support rules in UML class diagrams we extend the UML metamodel with a concept of a rule. The request for a UML-based rule modeling tool for the Semantic Web comes from the industry. Many companies claim that even if they understand benefits of using Semantic Web technologies like ontologies and rule languages, it is difficult for them to start since ontology architects and rule experts are quite expensive. A UML-based rule modeling approach for the Semantic Web will facilitate the use of the Semantic Web technologies by traditional UML modelers. The actuality of the proposed modeling approach also comes from the rules standardization efforts of W3C (http://www.w3.org/2005/rules/) and OMG (http://www.omg.org), which need rules modeling methodologies and tools.

In Section 2 we give a short overview of existing rule modeling approaches. In Section 3 we describe a metamodel and modeling language for derivation rules. In Section 4 we describe our rule modeling plugin for Fujaba, called Strelka and discuss modeling examples in Section 5. In Section 6 we summarize our results and describe future plans.

2. RELATED WORKS

The Protege tool provides facilities for ontology and rules modeling. In particular, it supports modeling in RDF [9] and OWL [8] as well as modeling of SWRL [3] rules. Protege requires a significant knowledge of ontology modeling. Moreover it is doubtful that it can be easily adopted in enterprizes, which already use UML technologies for software engineering.

There are ontology language specific tools for visual representation of ontologies, for instance, SemTalk from Semtation GmbH, which provides a visual language for modeling OWL ontologies. The approach of defining visual language for a particular ontology language has a lack of flexibility and scalability (since they provide visual notation for only one ontology language, for instance, OWL), while our UML-based approach has a power of MDA and allows obtain rules in language-independent manner.

There is a work on defining UML profile for ontologies and rules[10]. The approach defines a UML profile for SWRL rules and can be used for modeling of OWL ontologies and SWRL rules. Our approach is more general, since it supports not only SWRL-like (integrity and derivation) rules, but production rules and reaction rules as well. In addition, we provide special visual notation for rules, which cannot be obtained by the UML profile approach.

3. METAMODEL FOR RULES

In order to model rules with UML we have developed a UML-Based Rule Modeling Language (URML)[1]. This language supports modeling of derivation rules, production rules and reaction rules. In this paper we focus more on derivation rules, but briefly describe the general rule metamodel.

3.1 Rules

The general rule metamodel is depicted on Figure 1. A rule extends a UML TypedElement and belongs to a Namespace. We consider three main rule types:

[1]The URML on I1 website http://www.rewerse.net/I1 or in REWERSE I1 deliverable D8.

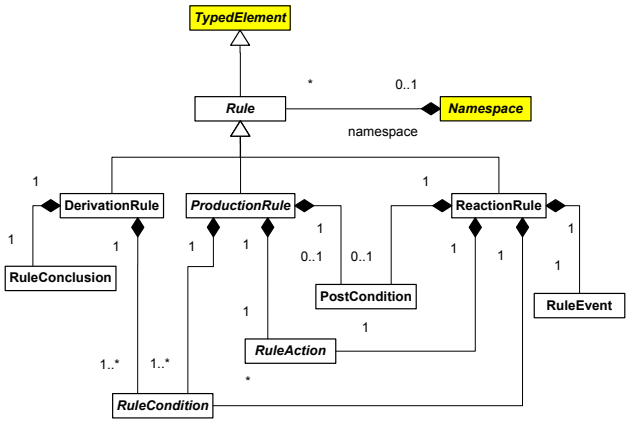

Figure 1: The URML rule metamodel

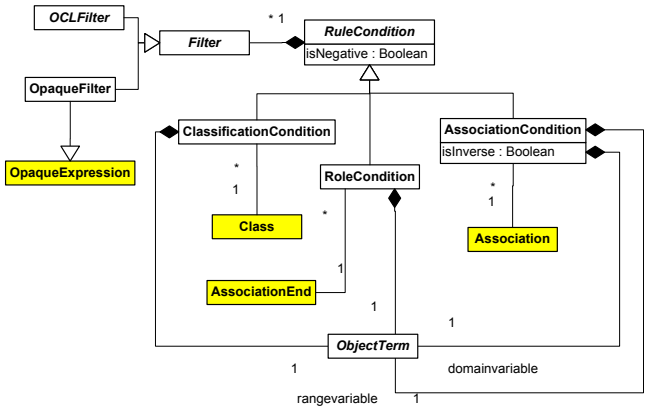

Figure 2: URML condition metamodel

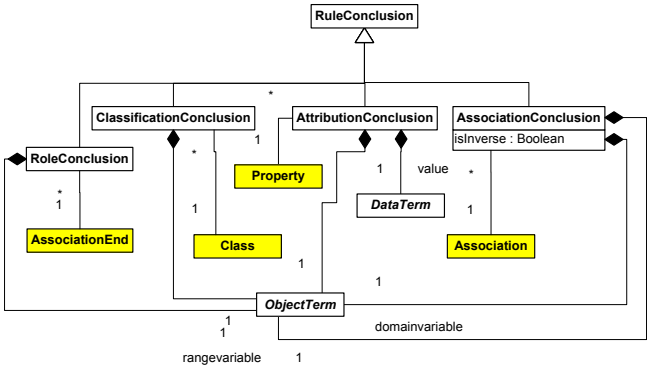

Figure 3: URML conclusion metamodel

DerivationRule has at least one condition and a conclusion. Such rule defines how model element can be derived. Example of derivation rule in the natural English language is: "*A gold customer is a customer with more than $1Million on deposit.*" This rule derives a concept of a *gold customer*, if condition "*more than $1Million on deposit*" is hold;

ProductionRule has at least one condition, one rule action and an optional post condition. Such rule performs an action if conditions are hold. For instance, the rule "*Exempt an investment from tax on profit if the stocks have been bought more than a year ago*" is a production rule, since if condition "*stocks have been bought more than a year ago*" is hold, then the action "*exempt an investment from tax on profit*" is performed;

ReactionRule may have several conditions, a triggering event, one rule action and an optional postcondition. Such rule formalizes event-condition-action behavioral model, where the action is executed on event with a condition satisfied. An example of such rule is "*When a share price drops by more than 5% and the investment is exempt from tax on profit, then sell it*". Event "*is share price drops by more than 5%*", condition is "*the investment is exempt from tax on profit*" and the action is "*sell it*".

3.2 Rule Conditions

A rule condition is a ClassificationCondition, RoleCondition or AssociationCondition (Figure 2).

ClassificationCondition refers to a UML Class, which is a condition classifier, and consists of an ObjectTerm, which is an object variable or an object;

RoleCondition refers to a UML AssociationEnd, which is a condition classifier, and consists of an ObjectTerm, which is an object variable or an object at the association end;

AssociationCondition refers to a UML Association, which is a condition classifier, and consists of two ObjectTerm's as a domain and a range, which are object variables or objects of classes from corresponding association ends of the Association.

A rule condition may have a filter expression. A filter is used to filter instances of a conditioned classifier. URML offers an option to express filters in OCL syntax, but an OpaqueFilter may be used for some vendor-specific implementation and syntax of filter expressions. Strelka tool supports OCL filters by integrating Dresden OCL Toolkit [7].

3.3 A Rule Conclusion

A rule conclusion is a RoleConclusion, ClassificationConclusion, AttributionConclusion, and AssociationConclusion (Figure 3). It is used in derivation rules in order to define a UML concepts (classes, associations, attribute values).

RoleConclusion refers to a UML AssociationEnd, which is a conclusion classifier, and consists of an ObjectTerm, which is an object variable or an object at the association end;

ClassificationConclusion refers to a UML Class, which is a conclusion classifier, and consists of an ObjectTerm, which is an object variable or an object;

AttributionConclusion refers to a UML Property and consists of an ObjectTerm, which is a context object or an object variable of the property and DataTerm, which is a value of the property;

AssociationConclusion refers to a UML Association, which is a conclusion classifier, and consists of two ObjectTerm's as a domain and a range object variable or object from corresponding association ends of the Association.

4. STRELKA - RULE MODELING FUJABA PLUGIN

In the REWERSE Working Group I1 we have developed a Strelka tool, which supports the URML. The tool is implemented as a Fujaba plugin. Following standard Fujaba architecture for writing plugins, it connects Fujaba metamodel with the URML metamodel, introducing required rule-related concepts like conditions, conclusions, rules, etc. We have used Fujaba to model URML metamodel, described in Section 3 and to generate corresponding Java sources. Below is a part of URML visual notation for rules, supported by Strelka:

Rules are represented as circles with identifiers;

Condition arrow refers to a *conditioned model element*, which is a classifier such as a class or an association. It may come with a *filter expression* selecting instances from the extension of the condition classifier and with an explicit object variable (or object variable tuple, in the case of an association) ranging over the resulting instance collection;

Negated condition arrow is crossed at their origin. It denotes a negated condition which has to be conjoined with one or more positive condition arrows such that its variables are covered by them;

Derivation rule is represented graphically as a circle with an internal label "DR" and a rule identifier attached to it. Incoming arrows represent conditions, outgoing arrows represent conclusions;

Conclusion arrow also refers to a classifier model element. Its meaning is to state that the predicate represented by the conclusion classifier applies to any instance that satisfies all rule conditions;

Filter expression is a text annotation of a condition arrow. Filter is used to filter instances of a conditioned classifier;

Variable is a text annotation of a condition or conclusion arrow and contains a name of an instance variable of a condition or a conclusion classifier.

A more detailed description of URML is available in the REWERSE Working Group I1 Deliverable D8.

In order to visualize the URML metamodel, we have implemented a rendering class for each URML metamodel class by extending Fujaba AbstractUnparseModule. In order to deploy modeled rules into a particular rule engine or rule-based application, Strelka supports serialization of rule models into the rule markup language R2ML[6], which is an acronym for the REWERSE Rule Markup Language. This rule language has been designed as a rule interchange format between different platforms and has such features as integration of functional languages (such as OCL) with Datalog languages (such as SWRL), the ontological distinction between objects and data values, the datatype concepts of RDF and user-defined datatypes, supports actions and events. Due to its design features, R2ML is an efficient intermediary format for rules and can be used as an interchange language between different rule systems and formalisms. There is a number of subsequent projects in the Working Group I1 dedicated to the rule interchange support between UML/OCL and OWL/SWRL using R2ML, Jess and R2ML, JRules and R2ML, etc. The R2ML serialization of URML models is already supported by Strelka. Since URML filter expressions (Figure 5) have OCL syntax, Dresden OCL Toolkit [7] and OCL for Fujaba have been adopted and integrated with Strelka.

5. MODELING EXAMPLES

In this section we provide two rule examples, modeled with the URML in the Strelka tool.

5.1 Deriving Association

Let's model the following derivation rule:

> If a rental car is stored at a branch, is not assigned to a rental and is not scheduled for service, then the rental car is available at the branch.

Its rule diagram is depicted on Figure 4, which is a screenshot of the Strelka plugin. The non-standard Fujaba modeling behavior, like arrows, which start and end in the middle of associations, is implemented in the tool. The rule is represented as a circle with an abbreviation DR, which stands for "Derivation Rule" and a rule identifier, which identifies the rule in a rule set. This rule has three conditions: *rental car is stored at the branch*, *rental car is not assigned to a rental*, *rental car is not a rental car scheduled for service*, which are visualized by incoming arrows, connecting conditioned classifier (a class or an association) with the rule circle. In order to visualize negated conditions, a condition arrow is crossed. The rule conclusion *the rental car is available at the branch* is visualized as an arrow from the rule circle to the derived association *isAvailableAt*. Condition and conclusion arrows are annotated with variables: **bra**, **rc**, **ren**. The semantics of the rule is captured by the following logical formula:

$$isAvailableAt(rc, bra) \longleftarrow isStoredAt(rc, bra) \wedge$$

$$\neg RentalCarSchduledForService(ren) \wedge$$

$$\neg \exists re(isAssignedTo(rc, re))$$

5.2 Deriving Class

Let's model the following rule:

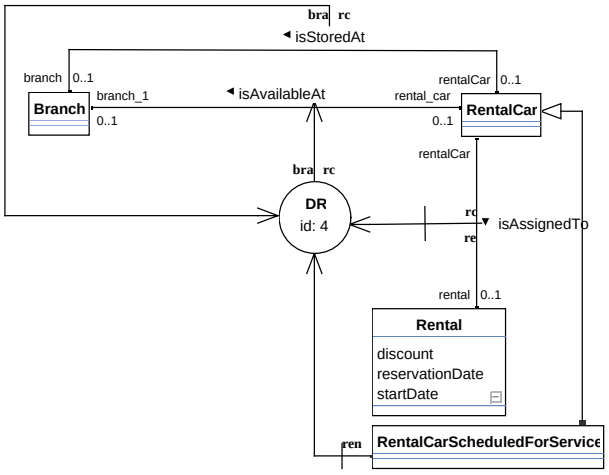

Figure 4: If a rental car is stored at a branch, is not assigned to a rental and is not scheduled for service, then the rental car is available at the branch.

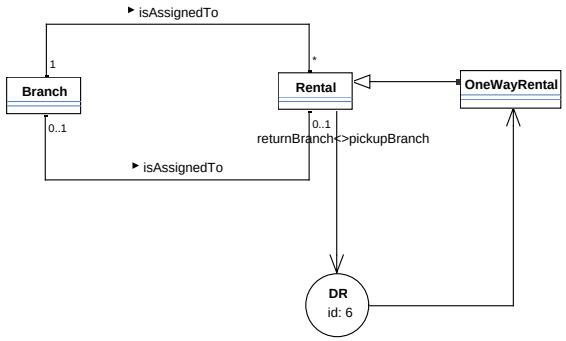

Figure 5: If return branch of a rental is different from pickup branch of a rental, then rental is a one way rental.

> If return branch of a rental is different from pickup branch of a rental, then rental is a one way rental.

Its diagram is depicted on Figure 5. This rule has only one condition with the filter expression in OCL syntax $returnBranch <> pickupBranch$.

6. EXPERIENCES AND CONCLUSIONS

In this paper we have introduced the Strelka tool for UML-based rule modeling. The tool can be used for visual modeling of different rule types. In order to support graphical notation of the URML we have added several new FSA Swing classes like double-head arrow, negated arrow, UML signal event, UML time event, etc.

From our experience, the main problem with Fujaba is a flexibility of the user interface. We have implemented several setting options in order to switch off/on methods and attributes visibility, method signatures and attribute types. The usage scenario assumes that the UML class diagram of a business vocabulary is developed first and then rules are visualized on subsequent diagrams. A subsequent diagram represents only a part of the business vocabulary, which participates in the rule. For instance, if only one of several associations between two classes participates in a rule, it is difficult to hide others from the diagram in Fujaba. One possible solution is to use Fujaba views. At the moment a modeler can choose an association for a view, but it is not shown, for instance, between which classes this association is.

During our work on URML models serialization into the R2ML we had problems with Dresden OCL toolkit integration with Fujaba. There is a project, which integrates OCL toolkit with Fujaba and we have based our R2ML serialization solution on it, but OCL support in Fujaba is still in alpha.

The future work on the tool includes integration of the rule verbalization component, which is currently under development in the Working Group I1. We are going to join Fujaba4Eclipse project and make Strelka working under Eclipse.

7. REFERENCES

[1] Ross, R. G., Principles of the Business Rule Approach. Addison-Wesley Information Technology Series (2003).

[2] Object Constraint Language (OCL), v2.0, http://www.omg.org/docs/ptc/03-10-14.pdf, last accessed Sep 4, 2006

[3] Semantic Web Rule Language (SWRL), http://www.daml.org/swrl, last accessed Sep 4, 2006

[4] Model Driven Architecture (MDA), OMG, http://www.omg.org/cgi-bin/doc?mda-guide, last accessed Sep 4, 2006

[5] W3C Workgroup on RIF Charter, http://www.w3.org/2005/rules/wg/charter, last accessed Sep 4, 2006

[6] Wagner, G., Giurca, A., Lukichev, S. (2006). A Usable Interchange Format for Rich Syntax Rules. Integrating OCL, RuleML and SWRL. In proceedings of Reasoning on the Web Workshop at WWW2006, May 2006.

[7] Dresden OCL Toolkit, http://dresden-ocl.sourceforge.net/, last accessed Sep 4, 2006

[8] Patel-Schneider, Peter F., Horroks I., OWL Web Ontology Language Semantic and Abstract Syntax, http://www.w3.org/2004/OWL, last accessed Sep 4, 2006

[9] Klyne G., Caroll J.J. (Eds.), Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C, 2004.

[10] Brockmans, S., Haase, P., Hitzler, P., Studer, R., A Metamodel and UML Profile for Rule-Extended OWL DL Ontologies., In proceedings of 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro, June 11-14, 2006, Springer Verlag.