Strelka - A Visual Rule Modeling Tool

Eclipse version 0.3 short manual

1 Introduction

The languages used in the communication between domain analysts and domain experts for analyzing and documenting system requirements should not be 'technical', but should allow visual and/or natural-language-like vocabulary and rule expressions that can be understood by domain experts without extensive technical training.

The UML offers a visual language for specifying vocabularies. Some rule types, for example, integrity constraints (invariants) and derivation rules can be represented in UML models by means of the Object Constraint Language (OCL). The OCL is a formal language, which is difficult to understand for people without a technical background.

In order to simplify rule modeling, the REWERSE Working Group I1 has developed

- a UML-based Rule Modeling Language (URML), which extends UML class models by adding rules, and
- Strelka, a tool for making URML models.

Strelka is available for download[1] in two versions: *i)* as a plug-in for the *Fujaba Tool Suite*, which is an open source UML case tool[2] and *ii)* as EMF-based Eclipse version.

2 URML – A UML-Based Metamodel and a Visual Notation for Rule Modeling

URML supports modeling of derivation rules, production rules and reaction rules. A rule is represented graphically as a circle with a rule identifier. Incoming arrows represent rule conditions or triggering events, outgoing arrows represent rule conclusions or produced actions.

Language Elements

Condition arrows refer 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 arrows are crossed at their origin; they denote a negated condition which has to be conjoined with one or more positive condition arrows such that its variables are covered by them.

Derivation rules are 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 arrows also refer 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.

[1] Strelka download page: http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/46
[2] Fujaba Tool Suite http://www.fujaba.de

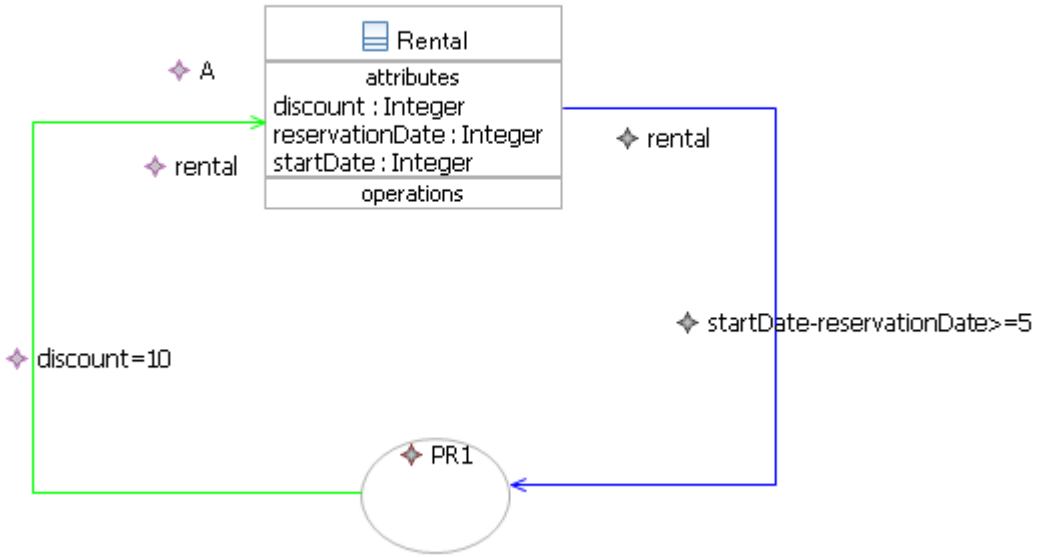

Figure 1: Production rule, assign action: If the reservation date of the rental is 5 days in advance of the reservation date, then give discount of 10 to the rental.

Production rules are represented graphically as a circle with an internal label "PR" and a rule identifier attached to it. Incoming arrows represent conditions, outgoing arrows with a double arrowhead represent actions.

Action arrows refer either to a class (in the case of a create, delete, assign or invoke action) or to an activity.

Reaction rules are represented graphically as a circle with an internal label "RR" and a rule identifier attached to it. There are two kinds of incoming arrows (condition arrows and event arrows) and two kinds of outgoing arrows (action arrows and postcondition arrows, both having a double arrowhead).

Event arrows may refer to a class.

A more detailed description of URML is available on the website of the Working Group I1[3].

3 Functionality of Strelka

3.1 Creating rule models

To create a rule, a user should create a new class diagram first. This can be done in the File → New → URML Diagram. All class diagram and rule modeling operations are accessible in the Palette on the right-hand side of the screen. **To create a class**, a user should click on the Class in the Palette and then click on the diagram. The class will appear and its name, attributes and operations can be specified .
To create a rule, a user should click on the Derivation rule or Production rule in the Palette and then on the diagram. The rule circle will show up and the rule id can be set.
To create rule condition arrow, a user should select a condition in the Palette. There is an association

[3]REWERSE Working Group I1 http://www.rewerse.net/I1

condition, which connects an association with the rule circle, a classification condition, which connects a class with the rule circle, and a role condition, which connects a role end with the rule circle. When the condition is selected in the Palette, click on the rule circle and drag the mouse to the condition classifier (to the class, association or role end). When the mouse button is released, a blue arrow from the condition classifier to the rule circle will show up.
To add a filter to a condition, a user should click on the small diamond near the condition arrow and enter filter text. A filter expression can also be entered in the Property area in the right-bottom corner of the applications. Click on the condition arrow to list and modify its properties.
To add a rule variable, a user should click on the small diamond near the beginning of the condition arrow and enter the variable name. The variable name can also be specified in the properties of the condition arrow in the right-bottom corner of the application.
To create a rule conclusion arrow, a user should select a conclusion classifier in the Palette, then click on the rule circle and drag the mouse to the conclusion classifier (class, association or a role name). When the mouse button is released, the red conclusion arrow will show up.
To create a production rule action, a user should select an action expression in the Palette. Four action types are supported: AssignVariableActionExpr is used to assign a value to a property, UpdateActionExpr is used to specify new values for object properties, RetractActionExpr is used to remove an object of the class, to which the action arrow points to, InvokeActionExpr is used to specify invocation of an operation, which belongs to the class, to which the action arrow points to. Visually, action types are distinguished by the capital character near the arrow head. **A** is used for the assign action, **C** for create, **D** for delete and **I** for the invoke action.

A number of rule examples come in the archive with the tool and is located in the examples folder.

3.2 Generating code for rule platforms

Strelka can be used in conjunction with I1 rule interchange service[4] to obtain a programming code of the rule for a specific platform. Currently, Jena 2, Jess, JBoss and F-Logic are supported as a target languages. In order to generate a rule code of the rule, right-click on the rule circle, select Translate to... and choose the target rule language. Strelka will connect to the web service and open the received code in the new window.

Two examples of derivation and production rules, created with Strela, are depicted on Figure 1 and 2. Figure 3 shows an excerpt from the Jess code, generated by Strelka for the rule from the Figure 1.

A user manual for Strelka is available on the website of the Working Group I1[5].

[4] I1 rule interchange service: http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/27
[5] REWERSE Working Group I1 http://www.rewerse.net/I1

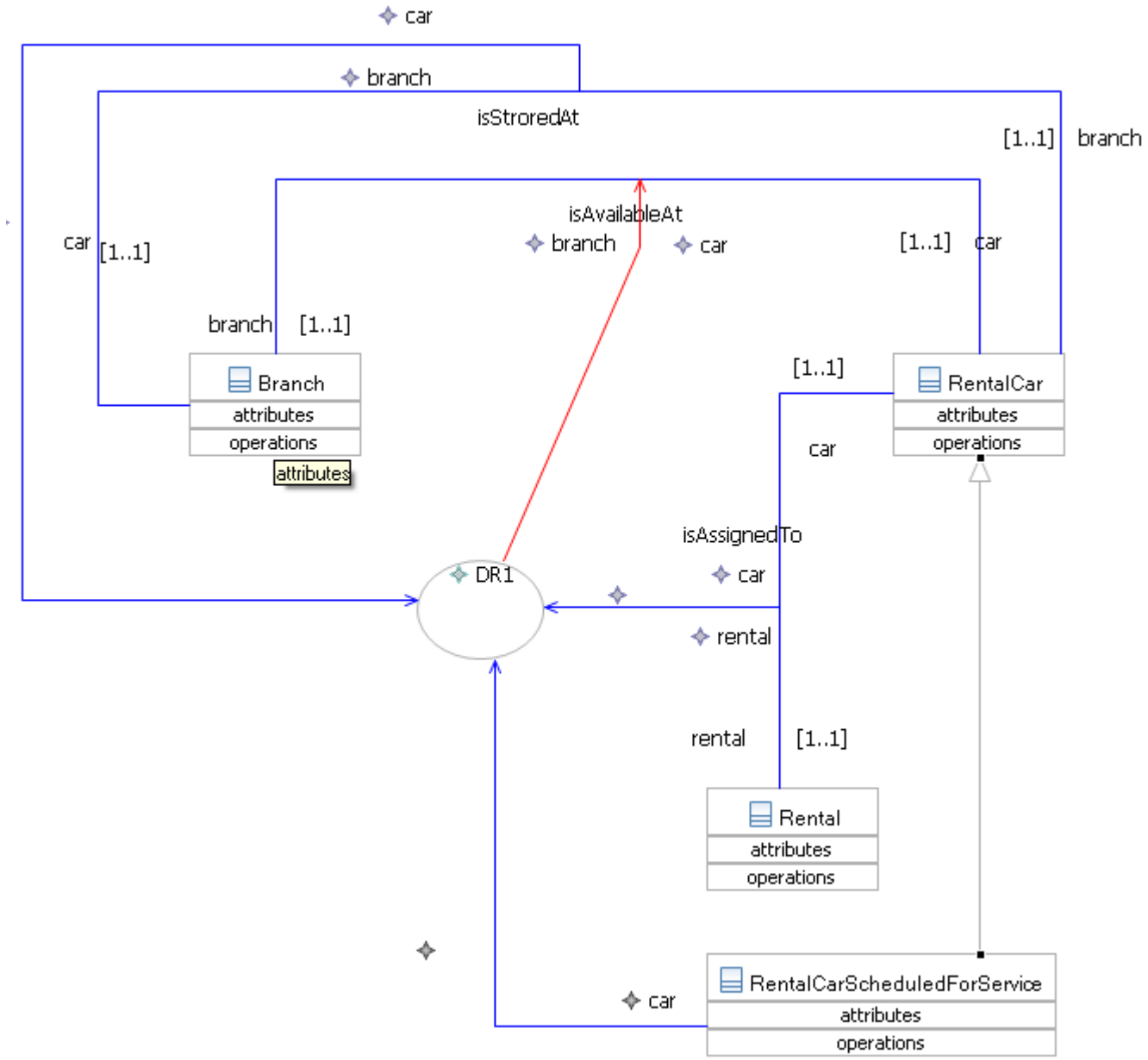

Figure 2: If a rental car is stored at a branch, is not assigned to a rental and is not scheduled for service, then the rental car is available at the branch.

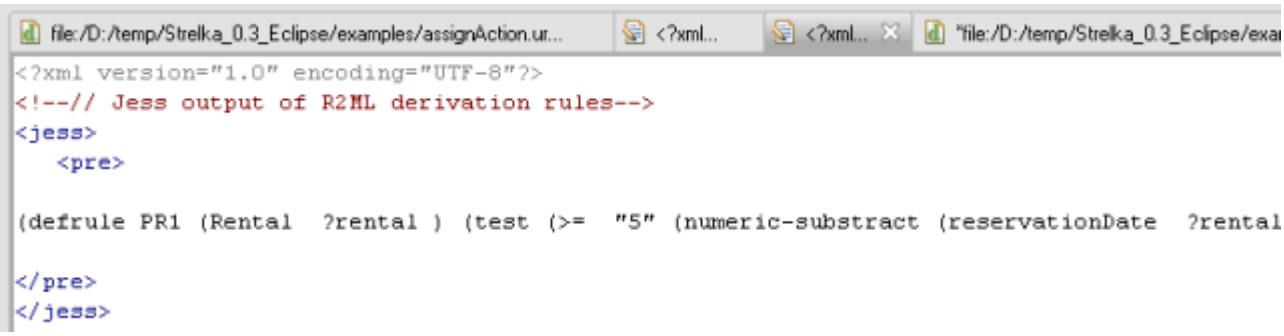

Figure 3: A preview of the rule from the Figure 1 in Jess.