Rule-Based Modeling of Semantic Web Services

Dragan Gasevic[2], Adrian Giurca[1], Sergey Lukichev[1], and Gerd Wagner[1]

[1] Institute of Informatics,
Brandenburg University of Technology at Cottbus, Germany
{Lukichev, G.Wagner, Giurca,}@tu-cottbus.de
[2] School of Interactive Arts and Technology,
Simon Fraser University Surrey, Canada
dgasevic@sfu.ca

A number of approaches to specifying Semantic Web Services have been proposed recently ([5], [1], [4]). In this paper we describe preliminary ideas of an alternative modeling approach, which uses a UML-based Rule Modeling Language (URML [6]) for defining behavior of Semantic Web Services. The URML supports modeling of Reaction Rules (also called Event-Condition-Action rules). A reaction rule consists of a *triggering event*, list of *conditions*, a *performed action* and an optional *postcondition*, which formalizes the state change after the execution of the action. This type of rules can be considered as a Web Service interaction description. The visual language serialize to the R2ML [8] markup language.

The approach under consideration is an evolutionary one and is compatible with a well-known language WSDL [9], which allows specifying web service interfaces in a platform-independent way. WSDL is an Interface Definition Language (IDL) and it needs to be the starting point in the Web Services Lifecycle. Hence, it should be properly modeled and implemented and needs the same respect as the other artifacts within the software system. And there needs to be some means of specifying WSDL in an easy way. However, creating a WSDL from scratch is not easy for most developers. WSDL modeling with UML-based language will simplify the interface definition process and is more usable than creating the XML representation from scratch.

Known solutions for ontology-based description of Web services are OWL-S [5] and WSMO [4]. However, these solutions are too revolutionary and do not try to be compatible with current W3C submissions regarding Web services description (i.e. WSDL), but it rather tries to bring a new platform.

On the other hand, there are other solutions that based on UML profiles that contain either elements of WSDL [2], [7] or OWL-S. Although, they are based on UML, they are still very low-level oriented, as they focus on implementation details covered either by WSDL or OWL-S.

To the best of our knowledge, existing specifications for describing and developing Web services have the following drawbacks:

- They are usually low level and platform-dependent;
- They are not directly built on domain rules;
- They are disconnected from types and domain concepts (e.g., "semantic" models);
- There are no standardized ways to define preconditions and postconditons of Web services behavior.

Since we also aiming at generation semantically enriched Web service descriptions, we propose using WSDL-S [1] that introduces a few WSDL extensions that fully address our goals, and yet it allows us preserving compatibility with the standard WSDL. WSDL-S has the following features:

- It is defined as an extension of the standard WSDL with the goal to be fully compatible with standard Web Service specifications and tools. So, it is an *evolutionary* approach rather than *revolutionary* approach such as OWL-S;
- It enables annotating datatypes using domain ontologies, but still they are defined by using XML Schema. In fact, there are new XML attributes added to the WSDL specification: *wssem:modelReference* to refer to the ontology concepts, and *wssem:schemaMapping* attribute defining how a complex type is transformed into a domain ontology specification by using a mapping language;
- *Precondition* XML element can be defined for each operation definition in WSDL. It defines a set of assertions that must be met before a Web service operation can be invoked;
- *Effect* XML element can be defined for each operation definition in WSDL. It states that the output is returned or it can make statements about what changes in the state are expected to occur upon invocation of the service;
- It is fully agnostic about the language used for defining domain ontologies (e.g., UML, ODM, OWL) and rules (OCL, SWRL, RuleML, R2ML).

As can be seen from the listed features of WSDL-S, this language has a lot of concepts, similar to the concepts of reaction rules, described above. A WSDL interface operation corresponds to a reaction rule in the following way:

- A WSDL *input* element is modeled as an incoming message that represents the triggering event;
- A WSDL *output* element is modeled as an outgoing message representing the triggered action;
- Rule conditions and the postcondition can be used to model the WSDL-S *precondition* and the WSDL-S *effect*;
- A WSDL *outfault* element can be modeled as an alternative action (in the form of an outgoing message) bound to a corresponding error condition in a reaction rule of the form *ON-IF-THEN-ELSE*.

The following reaction rule example [3] illustrates the above mappings:

This rule has a R2ML serialization[3] and this serialization maps into the following WSDL-S parts:

The *vocabulary* of this rule corresponds to the WSDL-S type declaration:

```
<types>
 <xs:schema>
  <xs:complexType name="ProductOrderRequest">
   <xs:sequence>
```

[3] see more examples on reaction rules on the official R2ML web page `http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/6`

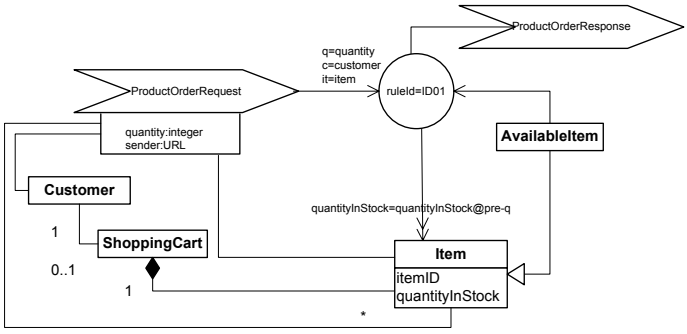

Fig. 1. On customer bookitem request, if the item is available, then approve order and decrease amount of items in stock.

```
    <xs:element name="item" type="srv:Item"
                            wssem:modelReference="b:Item"/>
    <xs:element name="quantity" type="xs:integer"
                               wssem:modelReference="b:quantity"/>
    <xs:element name="customer" type="srv:Customer"
                               wssem:modelReference="b:Customer"/>
    <xs:element name="sender" type="xs:anyURI"
                              wssem:modelReference="b:sender"/>
   </xs:sequence>
  </xs:complexType>
    <xs:complexType name="ProductOrderResponse">
   <xs:sequence>
    <xs:element name="validOrder" type="xs:boolean"
                                  wssem:modelReference="b:validOrder"/>
   </xs:sequence>
  </xs:complexType>
 </xs:schema>
</types>
```

The rule corresponds to the following WSDL-S interface fragment:

```
<interface name="PurchaseOrder">
 <operation name="b:processPurchaseOrder" pattern="wsdl:in-out"
         wssem:modelReference="b:RequestPurchaseOrder">
  <input messageLabel="processPurchaseOrderRequest"
             element="srv:ProductOrderRequest"/>
  <output messageLabel="processPurchaseOrderResponse"
              element="srv:ProductOrderResponse"/>
  <wssem:precondition name="BookIsAvailable"
   wssem:modelReference=
      "b:rules/r2ml:RuleBase/r2ml:ReactionRuleSet/
         r2ml:ReactionRule[@r2ml:ruleID='ID01']/r2ml:conditions">
  <wssem:effect name="DecreaseItemStock"
   wssem:modelReference=
```

```
    "b:rules/r2ml:RuleBase/r2ml:ReactionRuleSet/
        r2ml:ReactionRule[@r2ml:ruleID='ID01']/r2ml:postcondition"/>
 </operation>
</interface>
```

The *conditions part* of the rule, the bookitem is available (corresponding to the element `r2ml:conditions` in the R2ML serialization)maps to the to the WSDL-S annotation element `wssem:precondition` by the meaning of the value of attribute `wssem:modelReference`. This value is an XPath expression which allows the WSDL-S processor to find the content of the conditions part of the rule having `r2ml:RuleID="ID01"`.

The *postcondition* part of the rule, decrease amount of items in stock (corresponding to the element `r2ml:postcondition` in the R2ML serialization) maps to the to the WSDL-S annotation element `wssem:effect` by the meaning of the value of attribute `wssem:modelReference`. This value is an XPath expression which allows the WSDL-S processor to find the content of the postcondition part of the rule with `r2ml:RuleID="ID01"`.

The action part of the rule maps into the WSDL `operation` element as below:

- The `r2ml:operationID` attribute into the `name` attribute of the WSDL operation i.e. `name="b:processPurchaseOrder"`;
- The `r2ml:arguments` into a set of `input` elements of the WSDL `operation`;
- The `output` element of the WSDL operation is obtained from the rule vocabulary (corresponding to the *action type* we have in the rule);

References

1. Akkiraju R., Farrell J.,Miller J., Nagarajan M., Schmidt M.-T., Sheth A., Verma K. (2005) Web Service Semantics - WSDL-S, W3C Member Submission, 7 November 2005, Version 1.0, `http://www.w3.org/Submission/WSDL-S/`
2. Bezivin, J., Hammoudi, S., Lopes, D., Jouault, F., Applying MDA Approach for Web Service Platform, *In Proceedings of the 8th IEEE International Conference on Enterprise Distributed Object Computing Conference*, 2004, pp. 58-70.
3. Giurca, A., Lukichev, S., Wagner, G. (2006) Modeling Web Services with URML, *In proceedings of Semantics for Business Process Management Workshop*, Budva, Montenegro, June 2006.
4. Lausen H., Polleres A., Roman D. (Eds) (2005) Web Service Modeling Ontology (WSMO), W3C Member Submission, 3 June 2005, `http://www.w3.org/Submission/WSMO/`.
5. OWL-S: Semantic Markup for Web Services (2004) W3C Member Submission, 22 November 2004, `http://www.w3.org/Submission/OWL-S`.
6. URML: A UML-Based Rule Modeling Language (2006). On REWERSE Working Group I1 website: `http://oxygen.informatik.tu-cottbus.de/rewerse-i1/?q=node/7`
7. Vara J.M., de Castro V., Marcos E. (2005) WSDL Automatic Generation from UML Models in a MDA Framework, *International Journal of Web Services Practices*, Vol.1, No.1-2, 2005, pp. 1-12.
8. Wagner G., Giurca A., Lukichev S. (2005). R2ML: A General Approach for Marking up Rules, Dagstuhl Seminar Proceedings 05371, in F. Bry, F. Fages, M. Marchiori, H. Ohlbach (Eds.) *Principles and Practices of Semantic Web Reasoning*.
9. Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, W3C Candidate Recommendation, 27 March 2006, `http://www.w3.org/TR/wsdl20`