Model-driven Web Services Development

Roy Grønmo, David Skogan, Ida Solheim, Jon Oldevik
SINTEF Telecom and Informatics, Forskningsveien 1, N-0314, Oslo, Norway
{ roy.gronmo | david.skogan | ida.solheim | jon.oldevik }@sintef.no

Abstract

Web service technologies are becoming increasingly important for integrating systems and services. There is much activity and interest around standardization and usage of web service technologies. The Unified Modeling Language (UML) and the Model Driven Architecture (MDA)™ provide a framework that can be applied to web service development. This paper describes a model-driven web service development process, where web service descriptions are imported into UML models; integrated into composite web services; and the new web service descriptions are exported. The main contributions of this paper are conversion rules between UML and web services described by Web Service Description Language (WSDL) documents and XML Schema.

1. Introduction

Web services are functional components, available over the Internet, and described in the Web Service Definition Language (WDSL) [4]. This paper[1] investigates the use of UML to express the contents and behavior of web services in a more understandable way than WSDL.

UML modeling for automatic generation of CORBA IDL, Java code, EJB etc. has been successfully tried out (e.g. [10]). Model-driven approaches are also recommended by the MDA initiative of the OMG [2]. When it comes to web services, several UML enthusiasts have experimented with UML diagrams for automatic generation of WDSL service descriptions [5, 7]. This paper questions the WSDL-dependency of current service modeling approaches and investigates the usefulness of WSDL-independent UML models for web service specification. It raises the following two questions:

(a) Are WSDL-specific UML constructs necessary to understand what the web service does? Or does pure UML provide even better understanding?

(b) Are WSDL-specific UML constructs necessary for forward and/or reverse engineering of web services? Or can pure UML be used successfully for the same conversions?

The first question is discussed in Section 2, and the second in Section 3. The experiences with the conversion rules developed and the conclusions of the paper are given in Sections 4 and 5.

2. Modeling Web Services in UML

Before starting the argumentation, we need an overview of the context in which UML models of web services seem to be useful. In model-driven development we use models to describe business concerns, user requirements, activities, information structures, components and component interactions of a system. These models govern the system development in that they can be transformed to program code. In the case of web service development the models are transformed into the Web Service Description Language. A number of web services are now available and it therefore seems natural to reuse existing web services and thus aim at creating composite web services.

Figure 1 shows a UML activity diagram indicating the steps of model-driven web services development of composite web services. In the first step (Discover Web Services) the developer uses a web-browser, a registry client to search and discover candidate web services that may be used in the composite service. The output of this activity is a list of web service descriptions, represented as WSDL documents. To follow the model-driven philosophy the developer needs to import the necessary web service descriptions into UML by a reverse engineering transformation (Import Web Service

[1] This work has been funded by the European Union project IST-2001-37724, Adaptable and Composable E-commerce and Geographic Information Services (ACE-GIS)

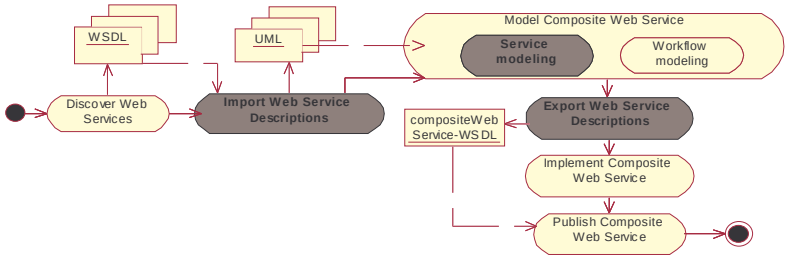

Figure 1: Steps of model-driven web services development

Descriptions). The output of this step is one or more UML models of the discovered web services.

The developer then uses a UML tool to review and integrate the imported models to form a model of a composite web service (Model Composite Web Service). This activity consists of two sub-activities: service modeling and workflow modeling, which focus on the interface of the service and its internal processes, respectively. The output is a new UML model representing the new composite web service with its services and its workflow. This model can now be used to generate the WSDL description of the composite service (Export Web Service Descriptions) and to generate the process description of the service that can be used to implement the service in for example in a business process execution engine (Implement Composite Web Service) [3]. Finally the service is published in an appropriate registry, making it available for use (Publish Composite Web Service).

Several authors have proposed WSDL-dependent UML profiles. Provost [5] has defined a UML profile for WSDL, introducing WSDL-dependent stereotypes. Gardner [3] takes a similar approach to workflow modeling (which is however outside this paper's scope), introducing a UML profile for BPEL4WS and conversion to BPEL4WS. Kollman et al. [8] give an overview of state of the art in reverse engineering, in which none of the referred tools uses platform-independent models. The Hypermodel tool of Dave Carlson [9] has the ability to import XML Schema (part of WSDL) into UML, but the resulting UML model will have extensions specific to XML Schema. Conversely, Thöne et al. [7] present platform-independent service and workflow modeling, but have not defined the conversion rules to any target platform.

A closer look into a WSDL-oriented UML profile [5] shows a number of UML extension mechanisms used to define:

- stereotypes for the specific WSDL and XML Schema types such as <<*wsdl:portType*>>, <<*wsdl:service*>> and <<*xs:complexType*>>; and
- tagged values for representing bindings, access URLs, etc.

According to this profile, a web service should be modeled using the specified UML stereotypes and tagged values, resulting in a WSDL-dependent model. This model would contain a number of WSDL details being irrelevant for understanding the semantics of the service. Rather, there is a risk of getting lost in implementation details. Especially when modeling complex web services, a pure conceptual view is very helpful to the modeler's comprehension. This paper recommends using solely the semantics of the UML language in order to enhance the understanding and efficiency of service modeling. The major advantage of WSDL-independent UML models are:

- The same model may be used as a basis for conversion to more than one target platform (WSDL, IDL, Java etc.), or to later versions of the same platform.
- The high-level, graphical models are easier to understand as they do not have all the technical details of the target platform.

The next Section gives an example of a simple web service modeled in WSDL-independent UML.

3. Conversion between UML and WSDL

Figure 2 shows a web service that is modeled independently of WSDL. The right-hand side of the figure shows the corresponding WSDL document. The WSDL document is simplified for clarity by leaving out a few elements and attributes as well as removing all the XML namespace information. The only non-standard stereotype introduced is <<BusinessService>>, which represents the component implementing the web service. Due to the limited space of this paper, it is not possible to go into the details of the conversion rules. However, the following description related to the figure should give an overview of the rules. The described web service is called "MyWebService" and realizes a web feature service interface [6] and a payment interface.

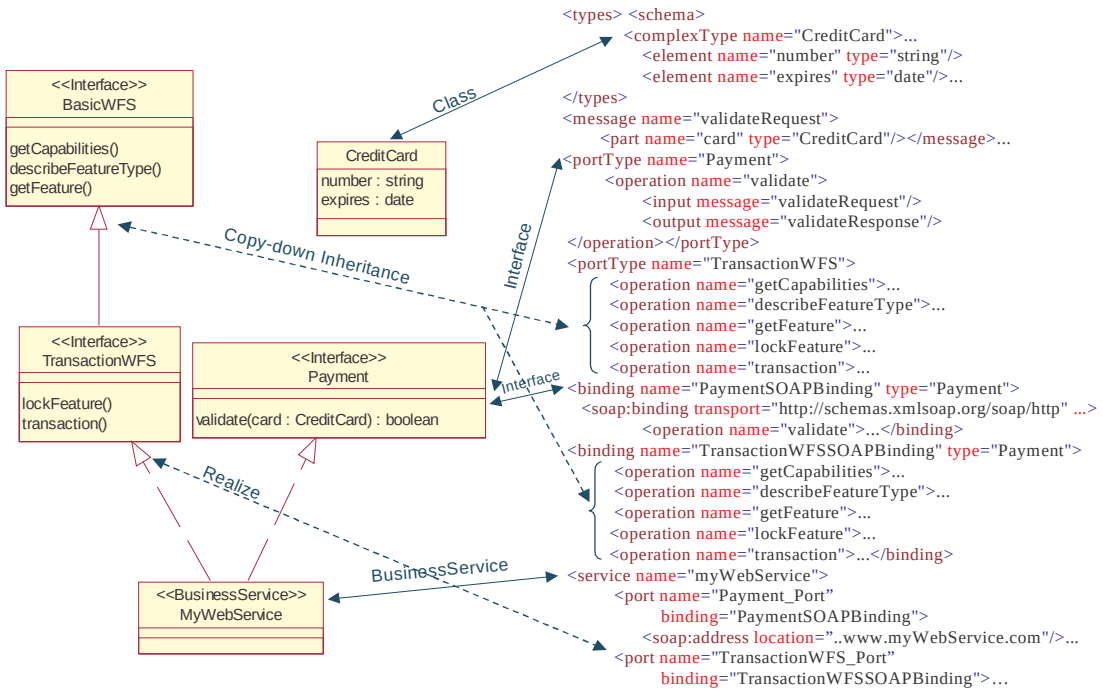

Figure 2: Conversion between a UML model and a WSDL document

The *CreditCard* class with the *number* and *expires* attributes, corresponds to a complexType in the type section of the WSDL file with the same name as the class, and part elements with the same names as the UML attributes. The *Payment* interface with the *validate* operation corresponds to both a WSDL portType and a WSDL binding. The *Payment* portType contains one WSDL operation for each interface operation. For each portType there must be at least one WSDL binding with type name equal to the portType name. Furthermore, the binding contains one WSDL operation for each operation of the interface. The binding information gives the choice of a specific protocol such as SOAP, HTTP GET/POST. This information is not present in our conceptual UML model.

The *TransactionWFS* interface provides operations for delivering geodata and possibly updating the geodata. This interface offers the operations of the OpenGIS consortium Web Feature Service Specification (WFS) [6]. *TransactionWFS* inherits three operations from the *BasicWFS* interface. UML interface inheritance results in copy-down of operations in WSDL, since WSDL does not support inheritance of portTypes or bindings. Class inheritance (not shown in the figure) is handled by XML Schema inheritance. This means that when reverse engineering from a WSDL document to UML, there will not be any interface inheritance.

The class called MyWebService is stereotyped as <<BusinessService>> and corresponds to a WSDL service of the same name. It realizes two interfaces, *TransactionWFS* and *Payment.* The two realized interfaces correspond to two WSDL ports. The two WSDL ports have a binding attribute equal to one of the bindings corresponding to the UML interface. The two ports are placed inside the WSDL service corresponding to *myWebService*. The resulting WSDL-service *myWebService* has six operations, corresponding to the operations of its realized interfaces.

4. Experiences

The conversion rules between UML and WSDL have been implemented in the UML Transformation Tool (UMT) [10]. This has enabled testing and practical experience of the conversion rules.

From UML to WSDL. Based on a simple UML model with a class which realizes some interface one may quickly come up with a WSDL document using the conversion rules. The transport protocol and encoding styles can be chosen at generation time and the protocols generated should follow recommendations of the Web Services Interoperability Organization [11]. The generation tool provides a good starting point and the resulting WSDL document can be further improved by a few manual insertions or corrections, e.g. by specifying necessary XML namespaces.

From WSDL to UML. Several WSDL documents found on the internet have been reverse engineered. However, even if the generated UML models are simple,

the quality of the models is not always good. The reason is that many WSDL documents do not have proper message or port type names. Another finding is that the WSDL documents that are generated from Java or .NET oriented WSDL-tools have different naming conventions. One example is a WSDL document where all operations have exactly one input parameter which is always named "parameters". This parameter is then of a type consisting of parts representing each actual parameter. It is clear that the semantics of the reverse generated UML model is dependent on the semantics included in the WSDL document.

5. Conclusions

This paper recommends developers to model web services as conceptual UML models without using WSDL-specific constructs. Further, it presents a to-ways mapping between a WSDL-independent service model in UML and the corresponding service description in WSDL. On the basis of practical tests, we claim that WSDL-independent UML models are better for understanding what a web service does; and that WSDL-independent UML models are sufficient for forward and/or reverse engineering of web services. The findings lead to the following conclusions:

(1) *A WSDL-independent UML model of a web service explains that service better than a WSDL-dependent model or pure WSDL does.* This because WSDL-specific UML constructs obscure rather than clarify the content and behavior of the web service. On the contrary, models ignoring the WDSL-specific information contain fewer technical details and are therefore easier to understand for humans.

(2) *WSDL-independent UML models simplify building of web services, especially when the services are complex.* This assertion is based on the fact that it is easier to integrate existing web services at the abstract level. The choice of protocols and bindings can be left to the underlying infrastructure.

(3) *Reverse engineering of WDSL specifications to WDSL-independent UML models works for all kinds of services.* The nature of the transformation rules allows us to reverse engineer any WSDL document. However, it does not always provide semantic useful models, due to the low semantic content of some WSDL documents.

(4) *Forward engineering (code generation) of WDSL-independent UML models into WSDL service descriptions works for all kinds of services.* The transformation rules generate valid WSDL documents based on a choice of predefined set of transport protocols and bindings. Hence we can in principle produce a web service specification for any UML class that has operations specified.

The recommended approach facilitates the inclusion of existing web services into a model-driven environment. It allows the developer to be more efficient in interpreting and reusing existing services in new settings.

References

[1] J. Oldevik and R. Grønmo, “ACE-GIS Deliverable D5.1.0 Basic Model-driven tool support,” 2002, www.acegis.net.

[2] OMG, 2002, "Object Management Group's Model Driven Architecture": www.omg.org/mda

[3] T. Gardner, “UML Modelling of Automated Business Processes with a Mapping to BPEL4WS,” presented at 17th European Conference on Object-Oriented Programming (ECOOP), Darmstadt, Germany, 2003.

[4] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, 2001, "Web Services Description Language (WSDL) 1.1, W3C Note": www.w3.org/TR/wsdl

[5] W. Provost, XML.com, 2003, "UML for Web Services": http://www.xml.com/lpt/a/ws/2003/08/05/uml.html

[6] OGC, “Web Feature Service Implementation Specification Version 1.0.0,” Open GIS Consortium Inc., OpenGIS Implementation Specification OGC 02-058 19 September 2002, http://www.opengis.org/techno/implementation.htm.

[7] S. Thöne, R. Depke, and G. Engels, “Process-Oriented, Flexible Composition of Web Services with UML,” presented at Int. Workshop on Conceptual Modeling Approaches for e-Business: A Web Service Perspective (eCOMO 2002), Tampere, Finland, 2002.

[8] R. Kollman, P. Selonen, E. Stroulia, T. Systä, and A. Zundorf, “A Study on the Current State of the Art in Tool-Supported UML-Based Static Reverse Engineering,” presented at Ninth Working Conference on Reverse Engineering (WCRE'02), Richmond, Virginia, 2002.

[9] D. Carlson, “Hypermodel”, www.ontogenics.com.

[10] SINTEF, “UML Model Transformation Tool”, http://umt-qvt.sourceforge.net.

[11] WS-I, 2003, "Web Services Interoperability Organization": http://www.ws-i.org/