

Transformations Between UML and OWL-S

Roy Grønmo¹, Michael C. Jaeger², and Hjørdis Hoff¹

¹ SINTEF Information and Communication Technology,
P.O.Box 124 Blindern, N-0314 Oslo, Norway
{Roy.Gronmo, Hjordis.Hoff}@sintef.no

² Technische Universität Berlin, Institute of Telecommunication Systems,
SEK FR6-10, Franklinstrasse 28/29, D-10587 Berlin, Germany
mcj@cs.tu-berlin.de

Abstract. As the number of available Web services increases there is a growing demand to realize complex business processes by combining and reusing available Web services. The reuse and combination of services results in a composition of Web services that may also involve services provided in the Internet. With semantically described Web services, an automated matchmaking of capabilities can help identify suitable services. To address the need for semantically defined Web services, OWL-S and WSMML have been proposed as competing semantic Web service languages.

Both proposals are quite low-level and hard to use even for experienced Web service developers. We propose a UML profile for semantic Web services that enables the use of high-level graphical models as an integration platform for semantic Web services. The UML profile provides flexibility as it supports multiple semantic Web service languages. Transformations of both ways between OWL-S and UML are implemented to show that the UML profile is expressive enough to support one of the leading semantic Web service languages.

1 Introduction

A growing number of Web services are implemented and made available internally in an enterprise or externally for other users to invoke. These Web services can be reused and composed in order to realize larger and more complex business processes. We define *Web services* to be services made available by using Internet protocols such as HTTP and XML-based data formats for their description and invocation. Web service registries such as UDDI allows for Web services to be published and discovered. A *Web service composition* is a description of how Web services can interoperate in order to perform a larger task.

The Web service proposals for description (WSDL [16]), invocation (SOAP [15]) and composition (BPEL4WS [2]¹) that are most commonly used, lack proper semantic descriptions of services. This makes it hard to search for appropriate services because a large number of syntactically described services need to be manually interpreted to see if they can perform the desired task. The requester may also need additional communication with the provider to determine the suitability of the service.

Semantically described Web services make it possible to improve the precision of the search for existing services and to automate the composition of services. Two recent

¹ BPEL4WS is currently being updated. It will be released in future under the name *WS-BPEL*.

proposals have gained a lot of attention; The American-based OWL Services (OWL-S) [4] and the European-based Web Services Modeling Language (WSML²) [17]. These emerging specifications overlap in some parts and are complementary in other parts. They are both described by low-level lexical notations. WSML uses its own lexical notation, while OWL-S is XML-based.

The leading organization for object-oriented programming, the Object Management Group (OMG), has defined the Unified Modeling Language (UML), a standard graphical language for expressing system development models. OMG also promotes a Model-Driven Architecture (MDA) approach for analysis, design and implementation of software systems. In a model-driven development process, models are used to describe business concerns, user requirements, information structures, components and component interactions. These models govern the system development because they can be transformed into executable code.

The work described in this paper adopts the MDA strategy for developing compositions of semantic Web services. An important question to be addressed in this paper is: *How can the new semantic Web service proposals be utilized within a model-driven Web service composition methodology?* An important part of the main question is to investigate if UML 2.0 [11] can be used as an integration platform for modeling semantic Web services. This implies the following requirements for the resulting UML models:

1. *Expressiveness.* They can contain sufficient semantic annotations so that they can be transformed to and from complete semantic Web service documents.
2. *Independence.* They are independent of the lexical semantic Web service languages.
3. *Readability.* They are easy to understand, interpret and specify for experienced modelers.

The motivation of the independence requirement is that one does not want to be tied to a particular semantic language, especially when there are competing standards. Furthermore, we believe that the semantic Web service developer can become more productive when working at a higher level than the low-level XML code. In order to satisfy the requirements, we have suggested a UML profile for semantic Web services and implemented transformations for both ways between UML and OWL-S. In addition, we explain how our UML profile can be transformed to WSML description, which demonstrates its semantic Web language-independence.

The paper is structured as follows: Section 2 briefly describes the procedure for semantic Web service composition that clarifies the need for a UML profile and transformations between UML and semantic Web service languages; Section 3 introduces the UML profile for semantic Web services; Section 4 presents the transformation rules both ways between OWL-S and UML; Section 5 shows the correspondence between WSML, OWL-S and our UML Profile; Section 6 evaluates our approach; Section 7 covers related work; and finally Section 8 concludes the paper.

² Note that WSML is often referred to as WSMO as it is defined by the WSMO working group and the WSML has only recently been published.

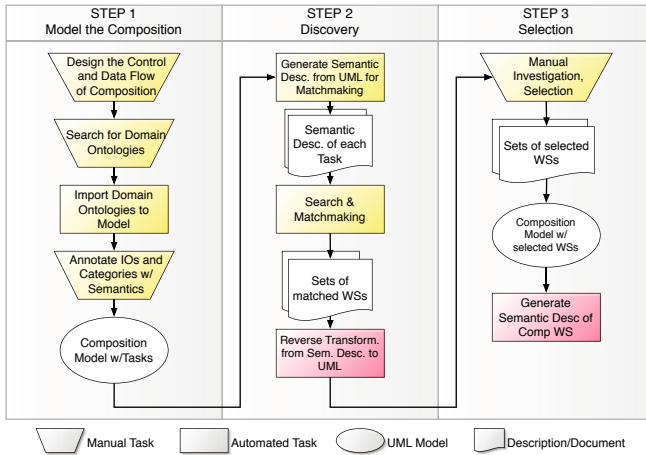


Fig. 1. Procedure for Composition of Semantic Web Services

2 Model-Driven Semantic Web Service Composition

This Section presents the model-driven procedure for creating compositions of semantically described Web services (Figure 1). The procedure consists of three main steps. The first step focuses on modeling a composition consisting of tasks, assuming that specific services are not yet identified. The second step focuses on service discovery and matchmaking based on the semantic description of tasks and services. In the third and final step, the service candidates are selected and the composition is completed. The procedure is described in a sequential manner although it should be emphasized that the procedure is an iterative process.

Step 1. The modeler specifies a new Web service composition in UML by creating a model that defines the control and data flow between the tasks. Each task is given with a task name and required inputs and outputs. Then, the designer must identify appropriate domain ontologies to semantically annotate the UML model. Domain ontologies may be searched at organizations or business interest communities. For example, the North American Industry Classification System represents such an effort³ and provides an ontology of products and services. After the modeler has determined one or more candidate ontologies, he imports these into the UML model. Such a technique has already been proposed by the work of Duric to bring OWL ontologies into a UML modeling environment [1]. Then, the desired Web services can be classified with inputs and outputs and a service category.

Only for the case that appropriate ontologies do not exist or are not suitable for being extended, the designer should consider to design a new ontology. However, we do not expect the need for creating a new ontology for the following reason: if a service

³ The ontology is available at <http://www.naics.com/>. An OWL version of this ontology is provided at <http://www.daml.org/ontologies/>.

requester cannot find ontologies, then he cannot presume that a provider provides a semantic description of his services using those non-existing ontologies. The outcome of the first step is a composition model that contains all the needed information for performing a discovery of service candidates.

Step 2. The second step handles discovery of suitable Web services. The discovery process is based on a matchmaking algorithm for semantic descriptions of services. It is assumed that a Web service registry is available with the following information provided for each Web service: *a*) a service interface description with location and *b*) a semantic description that can be used for the matchmaking process. The abstract composition model produced in step 1 is automatically transformed into a lexical document ("Semantic Desc. of each Task" in Figure 1) that can be parsed by a search and matchmaking algorithm. The recent proposals for matchmaking algorithms deal only with semantic matching of inputs, outputs and categories [9]. However, even if all these parts have clear matches, we are not guaranteed that the provided service is a perfect match. To guarantee perfect matches, further reasoning would be necessary that also take preconditions, postconditions and effects into account.⁴ Since this topic requires a lot of research, software tools and adoption of such techniques by the user community, we do not expect that all services can be discovered fully automatic in the near future. The matchmaking of inputs, outputs and categories will improve the precision of the search, but it will not remove the need for manual investigation of the discovered Web services to determine if they are suitable or not.

We anticipate that the found candidates provide a semantic description of their capabilities. The leading proposals for such semantic descriptions are OWL-S and WSML. However, the low-level and verbose OWL-S or WSML files are time-consuming and demanding to comprehend for the designer. To ease the manual investigation process, we propose to *reverse engineer* OWL-S and WSML into high-level UML models. Another benefit of importing the semantic description into UML, is that the imported services can be used directly as UML elements when finalizing the composition model. To accomplish this step, we have realized a transformation from OWL-S to UML which is further described in Section 4. The outcome of step 2 is a set of candidate services for each task.

Step 3. Based on the reverse engineered semantic description, the investigation of the services in UML can take place. The modeler selects the appropriate services and ideally at least one chosen service is assigned to each task. Tool support will preferably help the modeler to finalize the concrete composition model, which is the outcome of step 3. Theoretically, it could be wise to choose more than one service for a task. If during run-time a service becomes temporarily or permanently unavailable, an alternative service performing the same task can compensate the unavailable one.

At last, the concrete composition model is used to generate a semantic Web service description of the newly composed service (OWL-S, WSML etc.). This description can

⁴ The OWL-S proposal considers only the element *effects* to define resulting conditions from the service execution. The WSML proposal features both: it declares that postconditions cover the data output while effects can be used to describe general state changes not covering to the output.

Table 1. Summary of the UML profile

Stereotype	Extending UML Meta-model Element	Tagged Values	Usage
WebService	Activity	wsdl, service, port, operation	Used to model a single Web service operation. It's tagged values are sufficient to identify a web service operation
input	Activity inputPin		The stereotype is added to visualize if a pin is an input or an output. This is implicit when the WebService is part of a composition with incoming and outgoing flows, but is not visible when the WebService is visualized as a single service. It has an optional semantic type given by an association to a UOP OntClass.
output	Activity outputPin		<Same as for input>
Category	Comment	taxonomy, taxonomyURI, value, code	This item links the WebService to a category defined by a semantically defined concept within an ontology.
Pre-condition	Constraint		A pre-condition is of Constraint type and is visualized by a note in the diagram. A pre-condition is attached to all the input parameters included in the pre-condition statement. If no input parameters are part of the statement, then it is attached to the WebService item. The boolean pre-condition statement must be satisfied in order to execute the WebService.
Post-condition	Constraint		<Same as for Pre-condition with respect to output parameters instead of input parameters.> The boolean post-condition statement must be true when the WebService has terminated its execution.
Effect	Constraint		An effect statement defines the result of executing the WebService. The effect item is attached to the WebService

as opposed to a collection of operations. A **WebService** has four tagged values (**wsdl**, **service**, **port**, **operation**) which uniquely identify a Web service operation within a WSDL file. The other extensions to UML 2.0 activity models are introduced to support semantic annotation of WebService elements. A **WebService** can have an arbitrary number of **Input** and **Output** parameters. The **Input** and **Output** elements are minor extensions to the **inputPin** and **outputPin** of a UML **Activity**. The type of each of the parameters can be a syntactic type as previously for standard UML 2.0 **Pins**, but preferably now it will be a semantic type given as a **UOP OntClass**.

A semantic categorization of the **WebService** is given by a link to a **Category** element. The **Category** element extends the UML **Comment** element. It must be linked to a **WebService** and it has four tagged values which identify a category concept defined within an ontology. Although pre-conditions and post-conditions already exist in the UML meta-model, we have introduced two new elements for this purpose by extending the UML **Constraint** element. If a **Pre-condition** includes more than one input parameter in its expression, then it will be linked to all of the included parameters. The same rule applies to **Post-condition** with respect to output parameters. If the pre- or post-condition does not refer to any parameter, then it must be linked directly to the **WebService**. Note that the semantics of these new pre- and post-conditions are the same as in those already present in UML 2.0 by being conditions that apply to an operation (in this case specialised to be a Web service operation). The improvement is that they are now clearly linked to the **Pins** it concerns for improving the visualisation of the UML diagrams. Finally, the **Effect** element extends the UML **Constraint**. It is linked to a **WebService** to indicate the result of a successful execution of the **WebService**. The contents of **Pre-condition**, **Post-condition** and **Effect** should all be boolean expressions where the Object Constraint Language (OCL) is a natural candidate. Table 1 contains a summary of all the new elements in our UML profile. When modeling in the

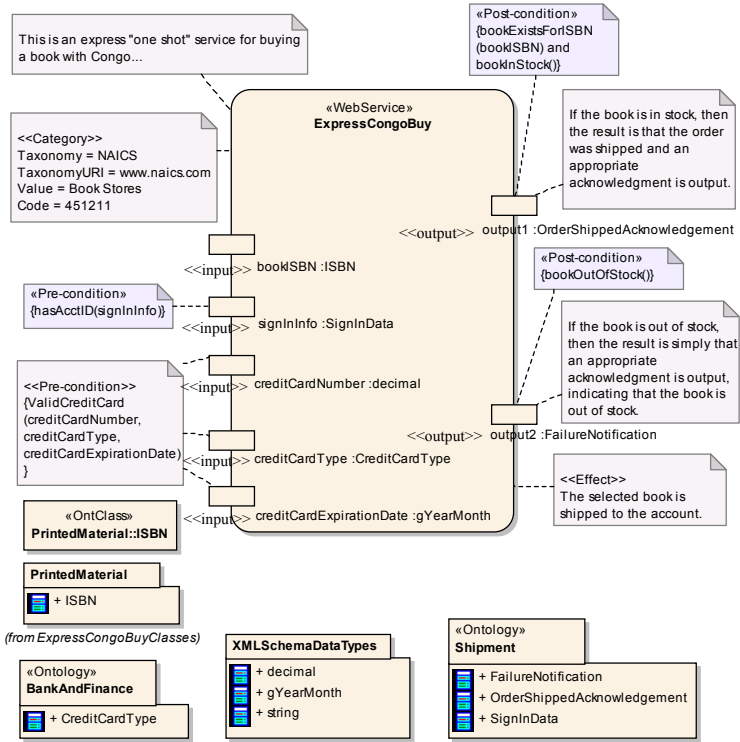


Fig. 3. ExpressCongoBuy service represented in our UML profile

UML profile it is important to note that also the reused elements from the UOP and the UML 2.0 activity models are available.

Now that we have defined how to model single, semantically annotated Web services in UML, we use UML 2.0 activity models to model compositions of semantically annotated Web services. The built-in control and data flow capabilities allow us to define how single semantic Web services interoperate in order to accomplish larger tasks. The resulting composition model can have its own boundary definition with a set of input and output parameters and thus can expose itself as a new Web service. A Web service may be decomposed as a composition, with some of its Web services being compositions itself and this may be recursively continued at an arbitrary number of levels.

3.1 The ExpressCongoBuy Example Expressed in the UML Profile

The model instance in figure 3 is used to explain our proposed UML profile by showing the OWL-S reference example ExpressCongoBuy in UML.⁵ ExpressCongoBuy is a Web service that allows a customer to buy books. In the ExpressCongoBuy example

⁵ The Congo example used can be found at the OWL-S home page: <http://www.daml.org/services/owl-s/1.1/examples.html>.

there are five input parameters to identify the customer information (`signInInfo`), credit card (`creditCardNumber`, `creditCardType` and `creditCardExpirationDate`) and the book (`bookISBN`).

There are two mutually exclusive output parameters in the example. The first output parameter indicates that the book is successfully purchased and shipped to the buyer's address, while the second output provides a message informing that the book was out of stock. The parameter types are linked to syntactic and semantic types. In the `ExpressCongoBuy` example the parameters `creditCardNumber` and `creditCardExpirationDate` are syntactically defined by referring to standard XML Schema data types. The other parameters are defined with semantic types as UOP OntClasses and grouped inside UOP Ontology packages.

Note that the model in figure 3 has only pseudo-logical expressions as the content of the pre-conditions, post-conditions and effect elements to make the example more readable. The post-conditions are attached to the output pins which states that there is a conditional output parameter. Only one of the two output parameters is returned, depending on which of the two post-conditions that evaluates to true. Finally we encourage the extensive use of human-understandable comments as UML notes attached to the relevant UML elements. This will help the model reader to interpret the model. All the three used plain comments (without stereotypes) in the UML model example are the original comments as found in the OWL-S example document.

4 Transformations Between OWL-S and Our UML Profile

The transformation between OWL-S and UML is necessary for two tasks when building semantic Web service compositions: *a)* to facilitate the reengineering process using the UML view, and *b)* to publish the semantically annotated composed service at the end. This Section provides an overview of our proposed transformation rules for *a)* and *b)*. Figure 4 shows a schematic view of the transformation elements. The figure consists of two parts: the left side shows the UML representation of the service. And the right side outlines fragments of the OWL-S description using a simplified non-XML-notation which is less verbose than the true XML. In between the two parts, the arrows indicate which part of the model corresponds to which part in OWL-S.

The service is represented in the UML model according to our UML profile as an activity with the stereotype `WebService`. Parameters of this activity element represent the inputs and outputs. The Web service has got five inputs and two (conditional) outputs. The figure also shows that properties of the service – such as the pre- and postcondition – are visualised with stereotyped notes. On the OWL-S side such an activity corresponds to the frame of one OWL-S document (indicated by the box labeled 1 in the figure). The inputs and outputs of an activity correspond to the `hasInput` and `hasOutput` elements in OWL-S accordingly (labelled 5 and 6).

In OWL-S it is proposed to use the Semantic Web Rule Language (SWRL [5]) for representing the `hasPrecondition`, `hasPostcondition` and `hasEffect` elements (labeled 4 and 7, please note that the figure does not show the postcondition due to space limitations). In UML, our proposal uses stereotyped notes containing an expression using OCL. However, the transformations of logical expressions would signifi-

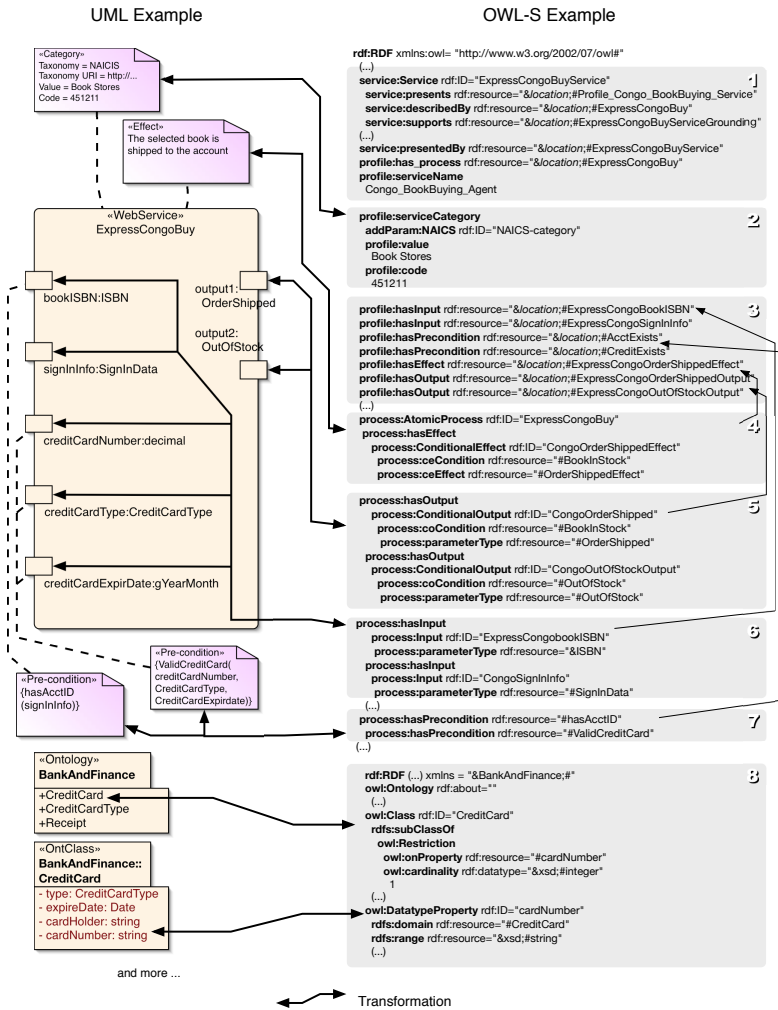


Fig. 4. Schematic view of transformations between UML and OWL-S

cantly extend the scope of the paper and thus is not handled by our transformations. The inputs, outputs, pre- and postconditions, and effects are generated at two places in the OWL-S document: the **Process** section and the **Profile** section. The transformation generates the elements in the **Process** part first. Then, these elements are basically duplicated for the **Profile** part. In fact, the referring elements found in the **Profile** section can be seen as a summary.

Reused ontologies are modeled in UML as separate packages with a URI as tagged value to identify the ontology. All such ontologies result in an import statement in the produced OWL-S document. Then the ontology concepts belonging to an imported ontology can be used at the adequate places (such as `parameterType`) by combining a short name of the imported ontology and the full name of the ontology concept. For

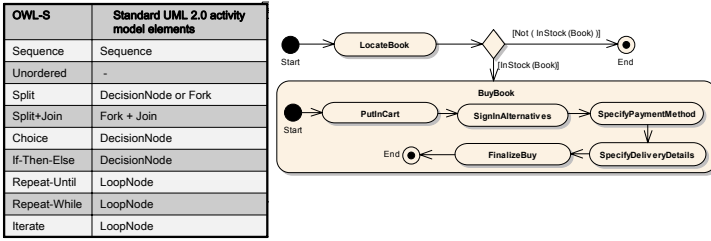


Fig. 5. FullCongoBuy service manually reverse engineered into UML

each new ontology concept a new owl:class is created. We do not explain further how to facilitate this part of the transformation, which is outlined with the box labelled 8, as this is covered by Duric’s work about representing OWL ontologies in UML [1].

The transformation can be extended to also handle conversations of Web services. A conversation occurs when several operations have to be called in a specific order before the service is completed. The OWL-S CompositeProcess element describes Web services that are to be called in a conversational manner. In OWL-S, the conversation of a Web service is described by a CompositeProcess element. The CompositeProcess represents the conversation model by using control flow constructs to connect other contained OWL-S Processes. An OWL-S CompositeProcess is a Web service composition which corresponds to a sub activity in UML. All the available control flow elements in OWL-S are shown in the table of Figure 5. The table shows corresponding UML elements for each of the OWL-S control flow constructions, except for the unordered element which have no natural corresponding element in UML. The unordered element contains a collection of Web services that shall be called in a sequence, but the order may be arbitrarily. A new stereotype called Unordered based on a sub activity could be introduced where the content is restricted to a set of unconnected activities. The single OWL-S Split+Join element corresponds to two separate elements in UML, a Fork followed by a join. Repeat-Until, Repeat-While and Iterate in OWL-S can all be mapped to UML LoopNode. However the LoopNode has no defined graphical notation, which then remains to be solved in order to have a visual representation. The rest of the mappings in the table are trivial.

The Congo example consists of two parts: ExpressCongoBuy and FullCongoBuy. The ExpressCongoBuy is a single Web service operation which we have already shown a representation of in our UML profile. The right part of Figure 5 shows how the conversational OWL-S FullCongoBuy example can be imported into a UML activity model by using the transformations defined in the table on the left side. Note that this is a simplified UML model which contains only the control flow and the individual service names. First the LocateBook service is called. The resulting output indicates if the book is in stock or not. The UML decisionNode control flow construct is introduced as a conditional OR-split that will finish the interaction if the book is not in stock. Otherwise, the composite service BuyBook is called, which involves a control flow graph of its own. A sequence of service calls is performed to put the book in cart, get the sign in alternatives, provide payment details, provide delivery information, and finally the complete Web service conversation ends.

5 Correspondence Between WSML, OWL-S and Our UML Profile

This Section explains the main parts of WSML and how this relates to OWL-S and our UML profile. WSML consists of four main parts: Ontologies, Goals, Mediators and Web services. The Ontologies part in WSML corresponds to OWL ontologies in an OWL-S document or to the UOP part of the UML profile. Goals represents abstract services for which one desires to find realizing Web services that can satisfy the requirements. There is no support for goals in OWL-S or in our specified UML profile. However, a new stereotype called Goal could be introduced which shares all the properties of the WebService concept except the four WSDL tagged values. Mediators deal with transformations and relationships that can be categorized as four kinds: OO-, WW-, GG- and WG-mediators. The OO-Mediator is the translation from one ontology to another. The WW-Mediator handles interoperability between Web services. The GG-Mediator expresses relationships between different goals. The WG-Mediator matches a goal with a realizing Web service. There is no equivalent to the mediators in OWL-S and its functionality is outside the scope of our UML profile.

The Web services part of WSML corresponds to our semantic Web service profile in UML and the OWL-S focus. The table on the left side of Figure 6 shows the correspondence between elements in the UML profile and the elements of WSML. The right side of the figure shows how the previously described ExpressCongoBuy can be expressed in WSML. We have simplified the WSML example by omitting the full definitions of the logical expressions for pre-condition, post-condition and effect. Many of the concepts have trivial mappings, but the definition of input and output parameters in the interface is quite different in WSML. The concepts to be used in the Web service interface are imported from ontologies and specialized by the WSML *subConceptOf* keyword. Namespaces are used to separate the imported super type from the new sub type. The sub type defines if the concept is used as an input or output parameter. The

UML Profile	WSML
WebService	webservice
input	imported concepts with mode hasValue <i>wsmI#in</i>
output	imported concepts with mode hasValue <i>wsmI#out</i>
pre-condition	precondition
post-condition	postcondition
effect	effect
Ontology (UOP)	ontology
OntClass (UOP)	concept

```

webservice http://.../ExpressCongoBuy
nonFunctionalProperties
  dc:title hasValue "ExpressCongoBuy"
  dc:creator hasValue "DERI International"
  dc:description hasValue "This is an express ..."
...
endNonFunctionalProperties
importedOntologies {<<http://.../Shipment>>,
  <<http://.../BankAndFinance>>,
  <<http://.../PrintedMaterial>>}
capability
  precondition <hasAcctID> + <ValidCreditCard>
  postcondition <bookExistsForISBN> + <bookOutOfStock>
  effect <Shipment of book>
concept bookISBN subConceptOf pm-namespace#bookISBN
nonFunctionalProperties
  wsmI#mode hasValue wsmI#in
  wsmI#grounding hasValue PrintedMaterial#ISBN
endNonFunctionalProperties
concept signInInfo ...
concept creditCardNumber ...
concept creditCardType ...
concept creditCardExpirationDate ...
concept output1 ...
nonFunctionalProperties
  wsmI#mode hasValue wsmI#out
  wsmI#grounding hasValue OrderShippedAcknowledgement
endNonFunctionalProperties
concept output2 ...
    
```

Fig. 6. Left: Mapping our UML profile to WSML. Right: ExpressCongoBuy in WSML

bookISBN parameter is shown with a complete WSML definition which illustrates the use of *subConceptOf*.

6 Evaluation

This Section evaluates the UML profile for semantic Web services against our requirements. The requirements which we have mentioned in Section 1 are:

- expressive enough for semantic Web services,
- independence of the lexical semantic Web service language, and
- readability for human readers.

Expressiveness. We have implemented a reverse transformation from OWL-S to UML and a forward transformation from UML to OWL-S. Both are implemented in the UML Model Transformation Tool (UMT) [7]. UMT is based on the XML Metadata Interchange Format (XMI) [10] which allows the realization of transformations that are UML tool-independent. The transformations between UML and OWL-S show to a large extent that our UML profile is expressive enough to capture and generate the needed semantic information of OWL-S. There are however three parts that are not implemented and that needs further extensions to make the profile fully expressive: metadata (such as contact name, address etc.), logic expressions and control flow. The support for organizational metadata about the service can be regarded as trivial. We have also suggested how the available control flow constructs can be handled, while logic expressions need more investigation to be covered properly.

The transformation from OWL-S to UML have been verified by testing on the Congo and Bravo OWL-S reference examples. The parser of a previously developed OWL-S matchmaking implementation [9] has successfully processed the OWL-S output of the UML to OWL-S transformation. However, we regard this test as a proof-of-work but not as a formal verification.

Independence. The independence of the lexical semantic Web service language cannot be fully claimed as we have not implemented any transformation between UML and the competitor WSML. However, we have explained how to fully support OWL-S and how to cover the relevant concepts of WSML in conjunction with our UML profile. Furthermore, the constructs we have proposed in our UML profile are not specifically designed to match one particular semantic Web service language.

Readability. The approach taken in this paper of using annotated UML activity models may be compared with a different approach of using UML interfaces with operations. UML Interfaces have the strength that they better define the separation between input and output parameters for which we needed to introduce stereotypes. We also need to be careful in the way the activity parameters are laid onto the activity depending on the incoming and outgoing flow when it is integrated into a composition model. If the input parameters are placed on the left side, then the incoming flow should also hit the left side boundary of the activity in order not to confuse a human interpreter.

A disadvantage of using UML interfaces is that comments, pre- and postconditions belonging to an input or output are not easily visualized in the diagram of an interface

element, while this is clearly visualized in our approach. The same argument goes for individual operations within an interface. These are handled properly in our approach since all operations are mapped to separate activities where comments, preconditions, postconditions, effects, category description etc. can be attached.

7 Related Work

We have already covered the two semantic Web service languages, OWL-S and WSML, that have the largest attention at the moment. Two other semantic Web service languages have also been proposed recently. WSDL-S [13] extends WSDL 2.0 with semantic descriptions. This language uses OWL types instead of the XML Schema syntactical data types to define the parameter types in operations. There are no extensions defined for expressing logical constraints such as pre-conditions, post-conditions and effects. Hakim-pour et al. [8] shows how The Operational Conceptual Modeling Language (OCML) is used by the IRS-III system to enable automatic service composition. They discuss differences and similarities between OCML, OWL-S and WSML. None of these two additional semantic Web service languages seem to introduce concepts that are not covered by our UML profile.

Duric explains in his work [1] how OWL concepts can be transformed into the UML Ontology Profile. This work is adopted as part of our transformation strategy from OWL-S to UML, and the domain ontology modeling part of this paper does not contain any new aspects. Elenius et al. [3] present the OWL-S editor as a graphical tool with the ability to both import and export OWL-S documents. Their approach is OWL-S-dependent, while our approach delivers a graphical language and transformations that can be reused towards different semantic Web languages, where OWL-S is just one candidate. Scicluna et al. [14] have a similar approach to ours by using UML 2.0 activity models to represent semantic Web services with a generation to OWL-S. Their approach is also tailored for OWL-S. Gomez-Perez et al. [6] have identified the same requirement as we have to deliver a semantic Web language-independent graphical language. To fulfill this need they propose the proprietary ODE SWS graphical language for modeling tasks that can be associated with inputs, outputs, preconditions and postconditions. In addition to their work we have defined transformation rules between OWL-S and UML. The inputs and outputs in ODE SWS are modeled as separate data objects on the outside of the task quite similar to UML 1.5 Activity models. In our approach we have used the more compact pin-notation of UML 2.0 Activity models that attaches a small symbol to the boundary, which makes the interface easier to read.

The METEOR-S tool presented by Rajasekaran et al. [13] is an approach where a user can annotate operations and its parameters with pre-conditions, post-conditions and semantic types in a tree view browser. From this tree view both WSDL-S and OWL-S generations have been implemented which shows a semantic language-independent approach. However, importing existing semantic Web service documents is not possible in this tool and compositions with control flow is not covered. Peer [12] proposes a lexical language to define imperative constructions with complex goals that are used to automatically generate executable Web service compositions. The executable Web ser-

vice compositions can be displayed with a graphical interface showing control flow and semantic annotations, but importing and exporting semantic Web service documents is not supported.

8 Conclusions

The contributions of this paper are a UML profile for semantic Web service composition and transformations both ways between UML and OWL-S. By importing semantic OWL-S descriptions of existing Web services into UML diagrams, we show that UML can be used as a common integration platform. The ability to generate Web service composition documents with semantic descriptions from a graphical model represents a valuable gain to the service developers, who otherwise have to write a lot of low-level XML code.

Fully defined transformations between UML and WSML is also desired so that our UML profile also can be used with the other leading semantic Web service specification. Another future improvement is to enhance the transformations between UML and OWL-S by also handling the logical expressions to cover the pre- and postconditions and the effects. This could be achieved by defining and implementing transformations between the logical languages used by OWL-S and WSML and the Object Constraint Language in UML. We see this as the next step towards providing a user-friendly environment to interpret and define the logical expressions.

Acknowledgements. The work of SINTEF is partially funded by the European IST-FP6-004559 project SODIUM (Service Oriented Development In a Unified framework). We would also like to thank Martin Hepp from DERI for his valuable input.

References

1. Dragan Djuric. MDA-based Ontology Infrastructure. *Computer Science Information Systems (ComSIS)*, 1(1):91–116, February 2004.
2. Satish Tatte (Editor). Business Process Execution Language for Web Services Version 1.1. Technical report, BEA Systems, IBM Corp., Microsoft Corp., <http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/>, February 2005.
3. Daniel Elenius, Grit Denker, David Martin, Fred Gilham, John Khouri, Shahin Sadaati, and Rukman Senanayake. The OWL-S Editor - A Development Tool for Semantic Web Services. In *2nd European Semantic Web Conference (ESWC 2005)*, Heraklion, Crete, Greece (accepted for publication), May 2005.
4. David L. Martin et al. Bringing Semantics to Web Services: The OWL-S Approach. In *Semantic Web Services and Web Process Composition, First International Workshop, SWSWPC 2004, Revised Selected Papers*, Volume 3387 of *Lecture Notes in Computer Science*, San Diego, California, USA, July 2004. Springer.
5. Ian Horrocks et al. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Technical Report, <http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/>, May 2004.
6. Asunción Gómez-Pérez, Rafael González-Cabero, and Manuel Lama. ODE SWS: A Framework for Designing and Composing Semantic Web Services. *IEEE Intelligent Systems*, 19(4):24–31, 2004.

7. Roy Grønmo and Jon Oldevik. An Empirical Study of the UML Model Transformation Tool (UMT). In *The First International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA)*, Geneva, Switzerland, February 2005.
8. Farshad Hakimpour, John Domingue, Enrico Motta, Liliana Cabral, and Yuanguai Lei. Integration of OWL-S into IRS-III. In *First AKT Workshop on Semantic Web Services, AKT-SWS04*, Athens, Greece, December 2004.
9. Michael C. Jaeger, Gregor Rojec-Goldmann, Gero Mühl, Christoph Liebetrueth, and Kurt Geihs. Ranked Matching for Service Descriptions using OWL-S. In *Kommunikation in verteilten Systemen (KiVS 2005)*, *Informatik Aktuell*, Kaiserslautern, Germany, February 2005.
10. Object Management Group (OMG). XML Metadata Interchange (XMI) Specification v1.2, OMG Document: formal/02-01-01. Technical report, January 2002.
11. Object Management Group (OMG). UML 2.0 Superstructure Specification, OMG Adopted Specification ptc/03-08-02. Technical Report, August 2003.
12. Joachim Peer. A PDDL Based Tool for Automatic Web Service Composition. In *Proceedings of the Second International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR)*, St. Malo, France, September 2004.
13. Preeda Rajasekaran, John A. Miller, Kunal Verma, and Amit P. Sheth. Enhancing Web Services Description and Discovery to Facilitate Composition. In *Semantic Web Services and Web Process Composition, First International Workshop, SWSWPC 2004, Revised Selected Papers*, volume 3387 of *Lecture Notes in Computer Science*, San Diego, California, USA, July 2004.
14. James Scicluna, Charlie Abela, and Matthew Montebello. Visual Modelling of OWL-S Services. In *Proceedings of the IADIS International Conference WWW/Internet*, Madrid Spain, October 2004.
15. World Wide Web Consortium (W3C). SOAP Version 1.2 Part 0: Primer. Technical Report, <http://www.w3.org/TR/soap12-part0/>, June 2003.
16. World Wide Web Consortium (W3C). Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. Technical Report, <http://www.w3.org/TR/wsdl20>, August 2004.
17. WSMO working group. D16.1v0.2 The Web Service Modeling Language WSML, WSML Final Draft. Technical Report, March 2005.