Bringing Semantics to Web Services: Model driven approach

DJAMEL AMAR BENSABER[1]
DJAMAL BENSLIMANE[2]
MIMOUN MALKI[1]
SIDI MOHAMED BENSLIMANE[1]

[1]Evolutionary Engineering and Distributed Information Systems Laboratory
Computer science Department, University of Sidi Bel Abbes
B.P 89 Sidi Bel Abbes, 22000 Algeria
[2]LIRIS Laboratory, University of Claude Bernard Lyon 69000, France
[1](Amarbensaber,Malki,Benslimane)@univ-sba.dz
[2]djamal.benslimane@liris.cnrs.fr

Abstract. The semantic web promises automates invocation, discovery and composition of web services by enhancing services with semantic descriptions. This paper describes a model driven approach to facilitate the construction of OWL-S specifications. The methodology is divided into three main steps. In the first step we reverse engineered WSDL documents into UML profile models that enable the use of high-level graphical models as an integration platform for semantic web services. In the second step, suitable domain ontologies are used for the semantic annotation of the UML models. Finally, in the third step a conversion tool will generate automatically the OWL-S description from these UML models. The UML profile provides flexibility as it can expresses multiple semantic web service concepts.

Keywords: WSDL, UML, MDA, OWL-S, Reverse engineering, Semantic web service.

(Received January 16, 2008 / Accepted May 26, 2008)

1 Introduction

Web services are gaining momentum as a new breed of distributed Web applications. Based on standard internet protocols such as SOAP , WSDL [2], and UDDI, Web services have become a key technology of B2B integration and service-oriented architecture. However, as the number of Web services increases and different people may describe similar or identical things differently, how to automatically process services at runtime is becoming an important issue. To make Web services more accessible to automated processes, there is a growing interest in bringing semantics into Web services. As a de facto standard, OWL-S [11, 12] defines a core set of markup language constructs for describing the properties and capabilities of Web services in unambiguous, computer-interpretable forms. OWL-S facilitates the automation of Web service tasks including discovery, execution, composition, and interoperation. However, the complexity of the OWL-S grammar makes it difficult to construct an OWL-S ontology manually and unfortunately, for the common developer, the learning curve for such languages can be steep, constituting a barrier to widespread adoption. Model Driven Architecture (MDA) recommended by the MDA initiative of the OMG [13, 14] is an approach to software development that is based on the creation of models rather than program code. The primary goals of MDA are portability, interoperability, and reusability through an architectural separation of concerns between the specification and implementation of software. MDA is dependent on and makes use of several other OMG standards, including UML, Meta-Object Facility (MOF), XML, Meta-Data Interchange (XMI) [17] and Common Warehouse Meta-model (CWM). In MDA based approaches, the focus is upon creation of software via the development of Unified Modelling Language (UML) models. UML [16] can be used as a convenient integration platform for modelling semantic web services since those corresponding UML models of web services haves the features :

1. Expressiveness. They contain sufficient semantic annotations to 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 easier to understand, interpret and specify for modellers.

In this paper we propose an approach to facilitate and raise the degree of automation in the semantic association process for the web services, we reverse engineer WSDL files into UML, annotate the generated models with relevant domain ontologies, and then we use a conversion tool to generate OWL-S descriptions from UML models. Our approach features the use of MDA concepts and strategies which enable the separation of the application design (or business logic) from the implementation platform. Specifically, these concepts allow us to translate XMI specifications (e.g., XML encodings of UML) into OWL-S via the eXtensible Stylesheet Language Transformations (XSLT) transformations [3]. We demonstrate our approach using CongoBuy example [11], and validate the correctness of our transformations using the OWL-S API [7]. The rest of the paper is organized as follows: In section 2, we describe the architecture and the main steps of our approach. Section 3 introduces the UML profile for semantic web service; where section 4 detail the reverse engineering process to generate UML model from WSDL files; section 5 shows the transformation rules between UML and OWL-S; Section 6 covers related work; and finally section 7 concludes the paper.

2 Our approach

This section describes our approach for using MDA techniques to synthesise OWL-S specification. The proposed methodology is broken up into three main steps (see Fig.1)

2.1 Reverse engineering process

The developer imports web service description into UML by a reverse engineering transformation. This activity consists of two sub activities: interface modelling and workflow modelling. Interface modelling specifies the service's interface and its operations, whereas workflow

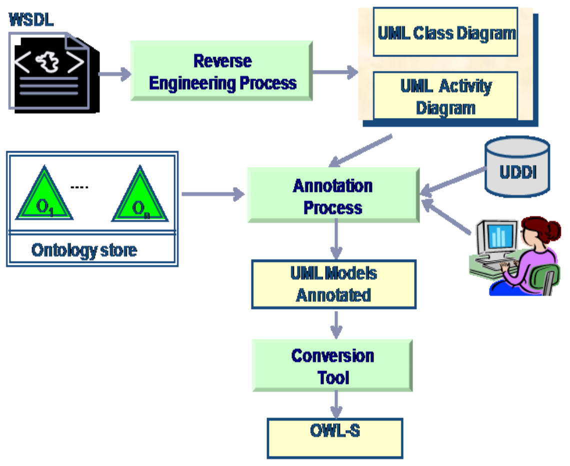


Figure 1: Main steps of our approach

modelling focuses on the internal behaviour of web service operation. Two UML diagrams will be generated from reverse engineering process; UML class diagram which describes the interface of service and UML activity diagrams of which the main part is obtained automatically when the order in which the activities will be performed is expressed by the developer by using data flow capabilities. The UML activity diagram is useful to describe the composite process of OWL-S process model. In this process, the WSDL file will be parsed and the information are extracted using JWSDL API combined with DOMSAX API to parse XML schema, then a set of transformation rules (see section 4) are applied to generate the corresponding elements in UML, these one will be exported in XMI format.

2.2 Annotation process

The generated activity UML models constitute the skeleton UML Models. The remainder of information will be enriched by as set of information from: UDDI, ontologies and the developer. Information in UDDI is useful to fill UML profiles part relating to owl-s profiles (category name, taxonomy, etc.). The "category" of service helps the designer to identify appropriate domain ontologies to semantically annotate the UML models. Domain ontologies may be fetched from organisations or business interest communities, e.g. the North American industry classification system represents such an effort and provides an ontology of products and services. After the modeller has determined one or more candidate ontologies, he imports these into the UML models.

Such a technique has already been proposed by Duric to bring OWL ontologies into UML modelling environment [4]. Ontology is visualized as UML interfaces, the XSD complex types presented as classes in UML are mapped to ontology concepts, the parameters, the operations, the pre-conditions, the post-conditions and the effects are also linked and annotated. The category is filled from the UDDI. Once UML profiles for semantic web services is completed with a little additional effort on behalf of the developer, the third stage of transformation to OWL-S descriptions can be launched.

2.3 Conversion tool

Conversion tool implements a forward transformation from UML models obtained in step 2 to OWL-S. The transformation process is based on the XSLT which is used in this project to convert an XML representation of an UML model into OWL-S description. The transformation rules rely on the UML profile we have created for OWL-S. The annotated UML models are used to generate a semantic web service description. This description can be automatically generated by the transformation which we have explained further in section 5. The UML diagrams are exported in XMI format. The conversion tool is invoked to run multiple transformations on the input file to produce the corresponding Service, ServiceProfile, ServiceModel and ServiceGrounding OWL-S documents. The transformations come in the form of XSLT transformations which automatically convert XMI input into OWL-S specifications. From the standpoint of MDA terminology, the process of creating a UML diagram from WSDL description can be considered equivalent to reversely engineering a PSM to PIM. The XSLT transformations correspond to "Other Information" necessary to generate a second-level PIM in the form of an OWL-S specification. The act of specifying groundings, which is part of the framework we are developing, corresponds to creating a PSM in the sense that the mapping provides information specific to create a specific executable implementation of a semantic service.

3 UML profile for Semantic Web Services

A UML profile is a collection of stereotypes, tagged values and custom data types used to extend the capabilities of the UML modelling language. We use a UML profile to model various OWL-S constructs in conjunction with the UML static structure diagram. In our approach, the stereotypes, tagged values, and data types serve to mark-up the platform-independent model, or PIM, in order to facilitate transformation to an OWL-S specification. Stereotypes work well to distinguish different types of classes and create a meta-language on top of the standard UML class modelling constructs. Tagged values allow the developer to attach a set of name/value pairs to the model. A set of name/value pairs is also a convenient way to attach information to the model which is needed in the transformation process. Fig.2 shows a meta-model of UML profile [6] where a group of UML extensions are introduced. UML profile is build on top of two existing metamodels which are shown as packages in Fig.2. The first reused meta-

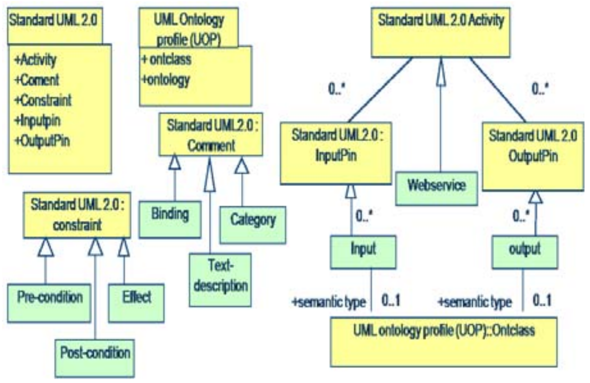


Figure 2: The meta-model of the UML profile

model is the UML Ontology Profile (UOP) that is more relevant when defining associations to our UML profile. UOP which is defined by Duric [4] is an extension to UML 1.5 class models which have been upgraded to UML 2.0 in [6]. UOP captures ontology concepts with properties and relationships. The UOP package contains the two elements (Ontology and OntClass). An OntClass element extends a UML class and represents a semantic concept. The OntClasses are semantically defined types which are grouped into Ontology packages. The second meta-model represents standard UML 2.0 activity model elements. The most central concept in the profile is WebService which extends the UML Activity element. A WebService represents a single callable Web service operation 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, and four other tagged values to specify access URIs necessary in grounding generating process like a URI for (wsdl service, wsdl port, wsdl version and wsdl document) . 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 tagged value indicates the access URI for WsdlInputMessage or WsdlOutputMessage is attached to each Input and Output parameters. The binding element extends the UML comment element, it is linked to WebService. This later defines the transport protocol to be used with its four associated tagged values (binding name, transport protocols, type and style). A semantic categorization of the WebService is given by a link to a Category element which extends the UML Comment element and it has four tagged values which identify a category concept defined within ontology. The pre- and post-conditions are linked to all of the included parameters in its expression. If the pre- or post-condition does not refer to any parameter, then it must be linked directly to the WebService. These new pre- and post-conditions 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 UML profile. Now that we have defined how to model single semantically annotated Web services in UML, we can 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 expose itself as a new Web service.

Table 1: Summary Of the UML Profile.

Stereotype	Extending UML meta model element	Tagged values	Usage
Web service	Activity	Wsdl, Service, Port, Operation, URI WSDL document, URI version WSDL, URI WSDL port, URI WSDL service	Used to model a single web service operation. Its tagged values are sufficient to identify a web service operation and his binding.
Input	Activity inputpin	Name Input URI WSDL Input message	The stereotype is added to visualize if a pin is an input or an output
Output	Activity outputpin	Name Output,URI WSDL Output message	Same as for <input>
Category	Comment	Taxonomy, TaxonomyURI, Value, Code	This item links the web service to a category defined by a Semantically defined concept within an ontology
Text-description	Comment		Provides a brief description of the service
Pre-condition	Constraint		A precondition is of constraint type and is visualised by a note in the diagram.
Post-condition	Constraint		Same as pre condition with respect to output parameters
Effect	Constraint		An effect statement defines the result of executing the web service
Binding operation	Comment	Binding name, style, type, transport	Used to identify binding operation

4 From WSDL to UML

Several authors have proposed WSDL-dependent UML profiles. Provost [20] has defined a UML profile for WSDL, introducing WSDL-dependent stereotypes. Gardner [5] takes a similar approach to workflow modelling, introducing a UML profile for BPEL4WS and conversion to BPEL4WS. Kollman in [10] give an overview of state of the art in reverse engineering, in which all the referred tools use platform-dependent models. The Hypermodel tool of Dave Carlson [1] 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. [23] present platform-independent service and workflow modelling, but have not defined the conversion rules to any target platform. In our approach, we have defined and implemented conversion rules from WSDL to UML to automate the reverse engineering process. We will investigate how to produce UML from WSDL by looking at an example web service with WSDL representation. The well-known "ExpressCongoBuy" provides a web service that allows customer to buy books. This section identifies two diagrams to modelling the WSDL content within UML: it's expresses a UML class diagram which describes the interface of web service and its operations, and UML activity diagrams which specifies the internal behaviour of web service operation. Notice that binding information can be left out at the modelling level of UML class diagram. However we introduce bindings and access URIs in UML activity diagram useful in the transformation process from UML to OWL-S grounding.

4.1 UML class diagram

Fig. 3 shows a WSDL-UML class diagram of "ExpressCongoBuy" service. The example has been modelled according to the WSDL independent model. The ExpressCongoBuy_service is mapped to UML class with stereotype "BusinessService". The service realizes one interface "ExpressCongoBuy_PortType". This interface has one operation which takes parameters as input and gives a boolean response indicate if the book is successfully purchased and shipped or not. These parameters refer to the two classes "CreditCard" and "SignInData". We briefly explain the main conversion rules for transforming a WSDL file into WSDL-UML class diagram.

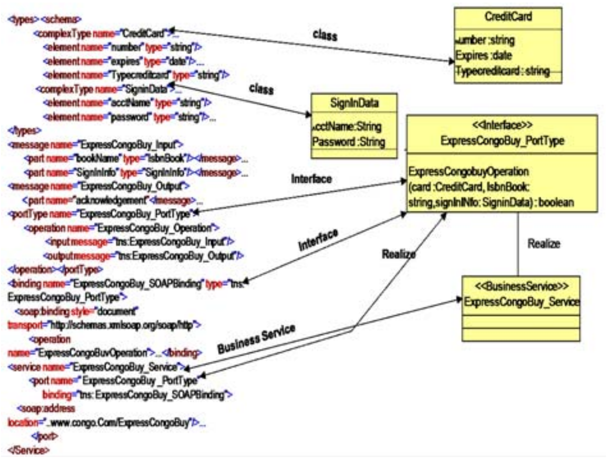
- A WSDL service is converted to a BusinessService.

- Each WSDL port within service is converted to realized relationship of the BusinessService.

- Each portType is converted to a UML interface.

- All operations contained in each port type are modelled as operations of the port type interface class, the parts name of input message become parameters of operation.

- Finally, all XML Schema defined types in the WSDL file are converted to UML classes.



```
<types> <schema>
    <complexType name="CreditCard">...
      <element name="number" type="string"/>
      <element name="expires" type="date"/>...
      <element name="Typecreditcard" type="string"/>
    <complexType name="SigninData">...
      <element name="acctName" type="string"/>
      <element name="password" type="string"/>...
</types>
<message name="ExpressCongoBuy_Input">
   <part name="bookName" type="IsbnBook"/></message>...
   <part name="SigninInfo" type="SigninInfo"/></message>...
<message name="ExpressCongoBuy_Output">
   <part name="acknowledgement"</message>...
<portType name="ExpressCongoBuy_PortType">
   <operation name="ExpressCongoBuy_Operation">
        <input message="tns:ExpressCongoBuy_Input"/>
       <outputmessage="tns:ExpressCongoBuy_Output"/>
</operation></portType>
<binding name="ExpressCongoBuy_SOAPBinding" type="tns:
ExpressCongoBuy_PortType">
  <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http">
     <operation
name="ExpressCongoBuvOperation">...</binding>
<service name="ExpressCongoBuy_Service">
     <port name=" ExpressCongoBuy _PortType"
        binding="tns: ExpressCongoBuy_SOAPBinding">
  <soap:address
location="..www.congo.Com/ExpressCongoBuy"/>...
      </port>
</Service>
```

Figure 3: WSDL- UML class Diagram

4.2 UML activity diagram

UML activity diagram specifies the internal behaviour of web service operation. In our approach each operation in port type is transformed to an activity. The built-in control and data flow capabilities allows developer to define the order in which the activities will be performed. The UML activity diagram is obtained from WSDL file by applying the following rules:

- Each operation in port type is converted to single web service operation stereotyped "WebService" with eight tagged values (WSDL, Service, Port, Operation, URI_WSDL_document, URI version WSDL, URI WSDL port, URI WSDL service) which uniquely identify a web service operation within a WSDL file.

- The part messages of input and output messages for operation are transformed to input and output parameters respectively stereotyped "Input" and "Output",

- The Access URIs for input and output message are modelled with tagged values associated to stereotyped "Input" and "Output".

- The binding operation information is presented as stereotyped note attached to WebService operation which extends the UML Comment element with four tagged values (BindingName, transport-protocols, style and type).

The UML activity diagram obtained after reverse engineering process represents a part of the diagram presented in Fig. 4. This diagram consists of a WebServiceOperation, the inputs and outputs parameters, stereotyped note for binding operation and three classes stemming from UML class diagram. These classes are mapped on existing OWL concepts during the annotation process of UML diagram. The rest of elements appearing in the diagram like category, pre-conditions, post- conditions and effects are defined and added in the annotation process.

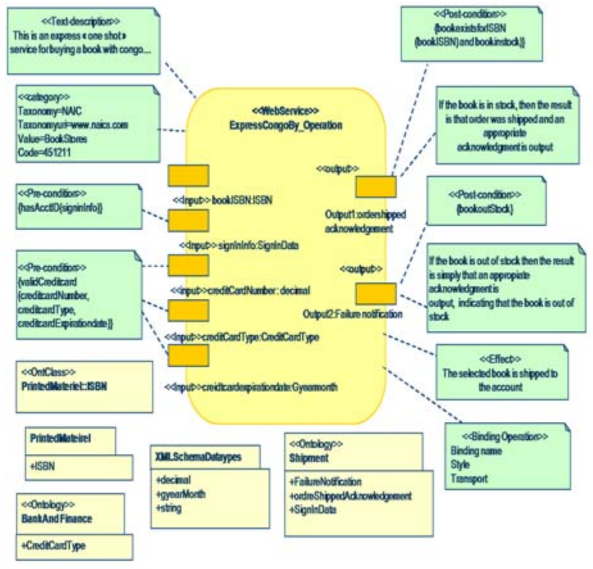

<<Text-description>>
This is an express «one shot» service for buying a book with congo...
<<category>>
Taxonomy=NAIC
Taxonomyuri=www.naics.com
Value=BookStores
Code=451211
<<Pre-condition>>
{hasAcctID(signinInfo)}
<<Pre-condition>>
{validCreditcard
(creditcardNumber,
creditcardType,
creditcardExpirationdate)}
<<OntClass>>
PrintedMateriel::ISBN
PrintedMateirel
+ISBN
<<Ontology>>
BankAndFinance
+CreditCardType
<<WebService>>
ExpressCongoBy_Operation
<<Input>> bookISBN:ISBN
<<Input>> signInInfo:SignInData
<<Input>>creditCardNumber: decimal
<<Input>>creditCardType:CreditCardType
<<Input>>creditcardexpirationdate:Gyearmonth
<<output>>
Output1:ordershipped acknowledgement
<<output>>
Output2:Failure notification
XMLSchemaDatatypes
+decimal
+gyearMonth
+string
<<Ontology>>
Shipment
+FailureNotification
+ordreShippedAcknowledgement
+SignInData
<<Post-condition>>
{bookexistsforISBN
(bookISBN) and bookinstock)}
If the book is in stock, then the result is that order was shipped and an appropriate acknowledgment is output
<<Post-condition>>
{bookoutStock}
If the book is out of stock then the result is simply that an appropiate acknowledgment is output, indicating that the book is out of stock
<<Effect>>
The selected book is shipped to the account
<<Binding Operation>>
Binding name
Style
Transport

Figure 4: ExpressCongoBuy service represented in our UML profile

4.3 Expresscongobuy example expressed in the UML profile

The model instance in Fig. 4 is used to explain our proposed UML profile by showing the OWL-S reference example ExpressCongoBuy in UML. ExpressCongoBuy is a Web service that allows a customer to buy books. In the ExpressCongoBuy example, 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. Notice that the model in Figure 4 has only pseudo-logical expressions as the content of the pre-conditions, post-conditions and effect elements. The post-conditions are attached to the output pins which state that there is a conditional output parameter. Only one of the two output parameters is returned, depending on which of the two post-conditions evaluates to true. A stereotyped note is attached to web service indicating the binding operation with tagged values (binding name, style and transport). The tagged values represented binding and access URI for input and output parameters are not presented in schema for lack of space, theses information are used in the transformation process for generating grounding.

5 Transformations between OWL-S and our UML profile

Fig. 5 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. Between the two parts, 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 labelled 1 in the figure). The inputs and outputs of an activity correspond to 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 [8]) for representing the hasPrecondition, hasPostcondition and hasEffect elements (labelled 4 and 7, please note that the figure does not show the post-condition due to space limitations).

In UML, our proposal uses stereotyped notes containing an expression using OCL. However, the transformations of logical expressions would significantly extend the scope of the paper and thus is not handled by our transformations. Inputs, outputs, pre- and post-conditions, and effects are generated at two places in the OWL-S document: Process section and 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 modelled 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 each new ontology concept a new owl: class is created. We do not further explain 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 [4].

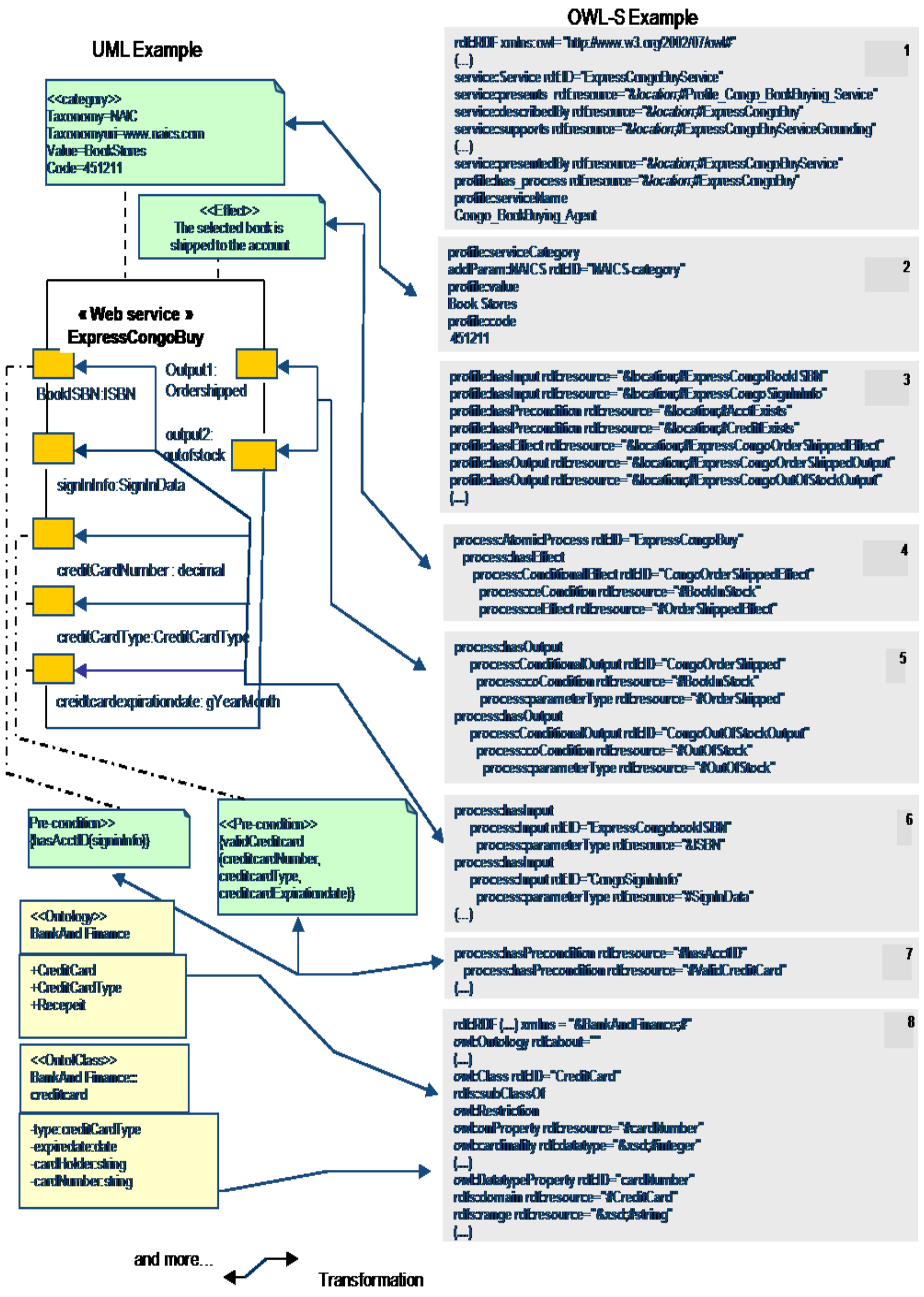
The transformation can be extended to also handle conversations of Web services. A conversation occurs when several operations have to be called in a specific order before the service is completed.

5.1 Grounding

A Service-Grounding consists of a number of AtomicProcessGroundings. These AtomicProcess Groundings contain mappings from Process types in an OWL-S specification to Port-Types in a WSDL specification. The AtomicProcessGrounding also contains mappings from parameters in an OWL-S specification to parameters in a WSDL specification. In our approach, the bindings and access URIs are transported from WSDL document into UML activity diagrams in the form of extensible elements (stereotypes and tagged values). So the process of generating the grounding is automatic by applying the following rules:

R1. Each AtomicProcess corresponds to one WSDL operation.

R2. As a consequence of the first rule, each input of an AtomicProcess is mapped to a corresponding message-part in the input message of the WSDL operation.



```
rdf:RDF xmlns:owl= "http://www.w3.org/2002/07/owl#"
(...)
service:Service rdf:ID="ExpressCongoBuyService"
service:presents  rdf:resource="&location;#Profile_Congo_BookBuying_Service"
service:describedBy rdf:resource="&location;#ExpressCongoBuy"
service:supports rdf:resource="&location;#ExpressCongoBuyServiceGrounding"
(...)
service:presentedBy rdf:resource="&location;#ExpressCongoBuyService"
profile:has_process rdf:resource="&location;#ExpressCongoBuy"
profile:serviceName
Congo_BookBuying_Agent
```

```
profile:serviceCategory
addParam:NAICS rdf:ID="NAICS-category"
profile:value
Book Stores
profile:code
 451211
```

```
profile:hasInput rdf:resource="&location;#ExpressCongoBookISBN"
profile:hasInput rdf:resource="&location;#ExpressCongo SignInInfo"
profile:hasPrecondition rdf:resource="&location;#AcctExists"
profile:hasPrecondition rdf:resource="&location;#CreditExists"
profile:hasEffect rdf:resource="&location;#ExpressCongoOrderShippedEffect"
profile:hasOutput rdf:resource="&location;#ExpressCongoOrderShippedOutput"
profile:hasOutput rdf:resource="&location;#ExpressCongoOutOfStockOutput"
(...)
```

```
process:AtomicProcess rdf:ID="ExpressCongoBuy"
  process:hasEffect
    process:ConditionalEffect rdf:ID="CongoOrderShippedEffect"
      process:ceCondition rdf:resource="#BookInStock"
      process:ceEffect rdf:resource="#OrderShippedEffect"
```

```
process:hasOutput
    process:ConditionalOutput rdf:ID="CongoOrderShipped"
      process:coCondition rdf:resource="#BookInStock"
       process:parameterType rdf:resource="#OrderShipped"
process:hasOutput
    process:ConditionalOutput rdf:ID="CongoOutOfStockOutput"
      process:coCondition rdf:resource="#OutOfStock"
        process:parameterType rdf:resource="#OutOfStock"
```

```
process:hasInput
    process:Input rdf:ID="ExpressCongobookISBN"
      process:parameterType rdf:resource="&ISBN"
process:hasInput
    process:Input rdf:ID="CongoSignInInfo"
      process:parameterType rdf:resource="#SignInData"
(...)
```

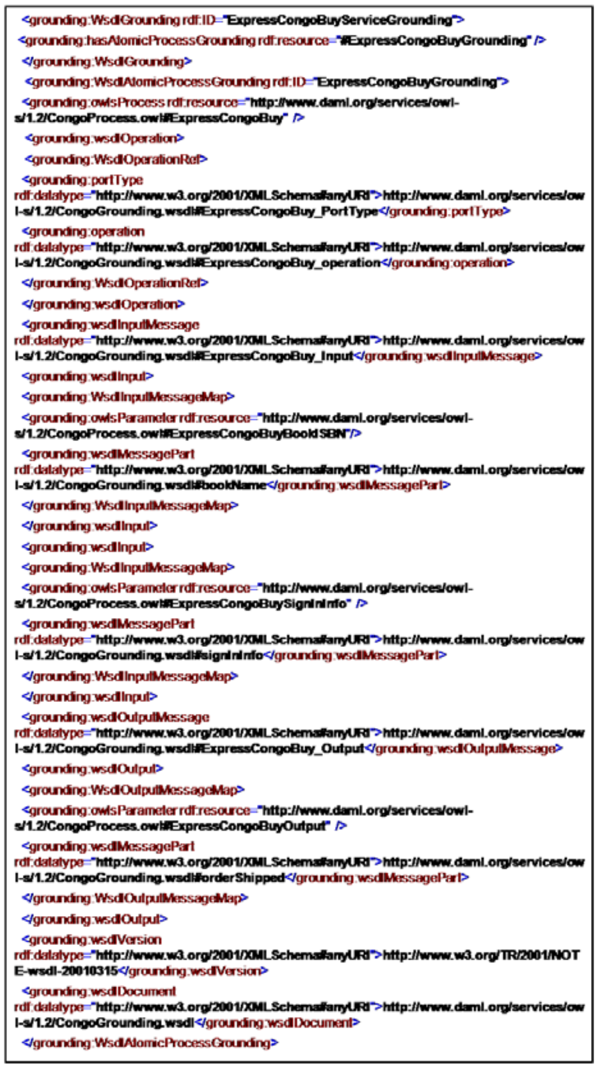
```
process:hasPrecondition rdf:resource="#hasAcctID"
  process:hasPrecondition rdf:resource="#ValidCreditCard"
(...)
```

```
rdf:RDF (...) xmlns = "&BankAndFinance;#"
owl:Ontology rdf:about=""
(...)
owl:Class rdf:ID="CreditCard"
rdfs:subClassOf
owl:Restriction
owl:onProperty rdf:resource="#cardNumber"
owl:cardinality rdf:datatype="&xsd;#integer"
(...)
owl:DatatypeProperty rdf:ID="cardNumber"
rdfs:domain rdf:resource="#CreditCard"
rdfs:range rdf:resource="&xsd;#string"
(...)
```

Figure 5: Schematic view of transformations between UML and OWL-S

Similarly for outputs, each output of an AtomicProcess is mapped to corresponding message-part in the output message of the WSDL operation.

R3. The type of each WSDL message part can be specified in terms of an OWL-S parameter (i.e., an XML Schema data type or an OWL concept). An excerpt of generated grounding is presented in Fig. 6.



```
 <grounding:WsdlGrounding rdf:ID="ExpressCongoBuyServiceGrounding">
<grounding:hasAtomicProcessGrounding rdf:resource="#ExpressCongoBuyGrounding" />
 </grounding:WsdlGrounding>
  <grounding:WsdlAtomicProcessGrounding rdf:ID="ExpressCongoBuyGrounding">
 <grounding:owlsProcess rdf:resource="http://www.daml.org/services/owl-
s/1.2/CongoProcess.owl#ExpressCongoBuy" />
  <grounding:wsdlOperation>
  <grounding:WsdlOperationRef>
 <grounding:portType
rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://www.daml.org/services/ow
l-s/1.2/CongoGrounding.wsdl#ExpressCongoBuy_PortType</grounding:portType>
 <grounding:operation
rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://www.daml.org/services/ow
l-s/1.2/CongoGrounding.wsdl#ExpressCongoBuy_operation</grounding:operation>
 </grounding:WsdlOperationRef>
 </grounding:wsdlOperation>
 <grounding:wsdlInputMessage
rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://www.daml.org/services/ow
l-s/1.2/CongoGrounding.wsdl#ExpressCongoBuy_Input</grounding:wsdlInputMessage>
 <grounding:wsdlInput>
 <grounding:WsdlInputMessageMap>
 <grounding:owlsParameter rdf:resource="http://www.daml.org/services/owl-
s/1.2/CongoProcess.owl#ExpressCongoBuyBookISBN"/>
 <grounding:wsdlMessagePart
rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://www.daml.org/services/ow
l-s/1.2/CongoGrounding.wsdl#bookName</grounding:wsdlMessagePart>
 </grounding:WsdlInputMessageMap>
 </grounding:wsdlInput>
 <grounding:wsdlInput>
 <grounding:WsdlInputMessageMap>
 <grounding:owlsParameter rdf:resource="http://www.daml.org/services/owl-
s/1.2/CongoProcess.owl#ExpressCongoBuySignInInfo" />
 <grounding:wsdlMessagePart
rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://www.daml.org/services/ow
l-s/1.2/CongoGrounding.wsdl#signInInfo</grounding:wsdlMessagePart>
 </grounding:WsdlInputMessageMap>
 </grounding:wsdlInput>
 <grounding:wsdlOutputMessage
rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://www.daml.org/services/ow
l-s/1.2/CongoGrounding.wsdl#ExpressCongoBuy_Output</grounding:wsdlOutputMessage>
 <grounding:wsdlOutput>
 <grounding:WsdlOutputMessageMap>
 <grounding:owlsParameter rdf:resource="http://www.daml.org/services/owl-
s/1.2/CongoProcess.owl#ExpressCongoBuyOutput" />
 <grounding:wsdlMessagePart
rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://www.daml.org/services/ow
l-s/1.2/CongoGrounding.wsdl#orderShipped</grounding:wsdlMessagePart>
 </grounding:WsdlOutputMessageMap>
 </grounding:wsdlOutput>
 <grounding:wsdlVersion
rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://www.w3.org/TR/2001/NOT
E-wsdl-20010315</grounding:wsdlVersion>
 <grounding:wsdlDocument
rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">http://www.daml.org/services/ow
l-s/1.2/CongoGrounding.wsdl</grounding:wsdlDocument>
 </grounding:WsdlAtomicProcessGrounding>
```

Figure 6: Excerpt of generated grounding

6 Related work

The OWL-S editor is a good standalone tool but is proprietary in nature and requires that the user become familiar with its user interface. Our approach leverages existing skills in UML modelling, which can greatly improve the efficiency of the semantic web service development workflow. Paolucci and al. developed the WSDL2OWLS system for automatically generating a one-shot OWL-S specification for a given WSDL file in [18]. As such, their approach is bottom-up technique and generates a complete grounding and incomplete profiles and process models. Our approach differs in that we use a hybrid approach, which combines between reverse engineering WSDL files into UML models and develop high- level OWL-S specifications. A similar method for creating a one-shot OWL-S specification is taken by Shen, Yang, Zhu and Wan in [22]. In their approach, a BPEL4WS specification is translated into an OWL-S specification. This technique generates a complete grounding and a complete process model using the BPEL composition operations as guidance. As a result, only the profile is incomplete, since no semantic information is used. The shortcoming of this approach is that it is limited to a single process instance and a single set of groundings. That is, it takes a bottom-up approach. Our approach takes a hybrid approach and thus supports development of more abstract OWL-S specifications. Jaeger, Engel and Geihs in [9] proposed a methodology for developing semantic descriptions of Web services using OWL-S. They recognize the lack of tool support for the development of semantic descriptions. A three-step process is introduced in which their tools will create a template using existing software artefacts (e.g, software models, WSDL), automate the identification of relevant ontologies and perform a classification based on those ontologies. The difference with our approach is that we discuss the development of OWL-S description in the context of an MDA environment. Gannod and Timm [24] have introduced a top-down approach, preferring to develop high-level OWL-S specifications and then using the flexibility of OWL-S groundings to map the OWL-S services to any number of potential WSDL realizations of an OWL-S process. The difference is that we reverse engineered WSDL files and we don't maps abstract OWL-S portions on concrete WSDL realizations, but we generate grounding form UML models built on imported WSDL descriptions. The Object Management Group (OMG) is creating a standard UML profile to support ontological development within UML tools via ontology definition metamodel [15]. The OWL profile for OMG supports generic OWL constructs but does not address the issue of OWL-S-specific constructs. For our work, creating a semantic web service specific UML profile turned out to be the only option to obtain the desired level of modelling granularity.

7 Conclusion and futures investigations

OWL-S provides an ontology for Web services that can be used to describe the semantics of a Web service. Unfortunately, adopting a language like OWL-S can be difficult because of the learning curve and current state of tool support. The recommended approach facilitates the specification of semantic web services using model driven development. By importing WSDL descriptions of existing web services into UML diagrams, we show that UML can be used as a common integration platform. The ability to generate semantic web service 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. Our UML profile is expressive enough to capture and generate the needed semantic information of OWL-S and also can be reused towards different semantic web languages such WSML or WSDL-S [21] which extends WSDL2.0 with semantic descriptions, so OWL-S is just one candidate. It is worth mentioning that in our current approach, the annotation of UML profile with real world ontologies (e.g the world-fact-book ontology contains more than 1100 concepts) can be very tedious task. To alleviate this problem (which has a major impact on the scalability of our system), we intend to automate further the annotation process by making usage of some matchmaking algorithms [19]. Some issues that will be resolved in future work include enhancing the transformations between UML and OWL-S by also handling the logical expressions to cover the pre- and post-conditions and the effects. This could be achieved by defining and implementing transformations between the logical languages used by OWL-S 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. The research issue of particular interest is that of automated composition of services. Currently, service composition is performed at design time. With semantic descriptions in place, automated composition is possible.

References

[1] Carlson, D. Hypermodel, www.ontogenics.com.

[2] Chinnici, M. J. R. C., E.R and Weerawarana, S. Web service description language 1.1. w3c note [online] available http://www.w3.org/tr/2007/rec-wsdl20-20070626.

[3] Clark, J. Xslt transformations v1.0. w3c recommendation [online] available http://www.w3c.org/tr/xslt (1999).

[4] Djuric, D. Mda-based ontology infrastructure. *Computer Science Information Systems*, 1(1):91–116, 2004.

[5] Gardner, T. Uml modelling of automated business processes with a mapping to bpel4ws. In *Proceedings of the 17th European Conference on Object-Oriented Programming (ECOOP)*, Darmstadt, Germany, 2003.

[6] Gronmo, J. M., R. and Hoff, H. Transformations between uml and owl-s. In *The European Conference on Model Driven Architecture -Foundations and Applications (ECMDA-FA)*, Nuremberg, Germany, 2005. Springer-Verlag.

[7] Group, T. M. Owl-s api. [online] available http://www.mindswap.org/2004/owl-s/api.

[8] Horrocks, I. e. a. Swrl: A semantic web rule language combining owl and ruleml. technical report, http://www.w3.org/submission/2004/subm-swrl-20040521.

[9] Jaeger, E. L., M.C and Geihs, K. A methodology for developing owl-s descriptions. In *Proceedings of the First International Conference on Interoperability of Enterprise Software and Applications Workshop on Web Services and Interoperability*, 2005.

[10] Kollman, S. P. S. E. S. T., R. and Zundorf, A. A study on the current state of the art in tool- supported uml-based static reverse engineering. In *Proceedings of the Ninth Working Conference on Reverse Engineering (WCRE'02)*, pages 22–32, Richmond, Virginia, 2002.

[11] Martin, e. a., D. Owl services coalition. owl-s: Semantic markup for web services, [online] available http://www.daml.org/services/owl-s/1.0/owl-s.pdf , 2003.

[12] Martin, e. a., D. Bringing semantics to web services with owl-s. In *World Wide Web*, pages 243–277, 2007.

[13] Miller, J. e. a. Mda guide version 1.0.1 (tech. rep. omg/2003-06-01). object management group, 2003.

[14] (OMG), O. M. G. Object management group's model driven architecture [online] available www.omg.org/mda.

[15] (OMG), O. M. G. Ontology definition metamodel (tech.rep. www.omg.org/docs/ad/05-01-01.pdf.

[16] (OMG), O. M. G. Uml 2.0 superstructure specification, omg adopted specification ptc/03-08-02. technical report, 2003.

[17] (OMG), O. M. G. Xml metadata interchange (tech. rep. formal/07-12-01), omg, 2007.

[18] Paolucci, S. N. S. K. . N. T., M. Towards a semantic choreography of web services: From wsdl to daml-s. In *Proceedings of the International Conference on Web Services*. IEEE, 2003.

[19] Patil, O. S. S. A. V. K., A. Meteor-s web service annotation framework. In *Proceedings of the 13th International world wide web conference, WWW2004*, pages 553–562, New York, USA, 2004.

[20] Provost, W. Uml for web services, http://www.xml.com/lpt/a/ws/2003/08/05/uml.html.

[21] Rajasekaran, M. J. V. K., P. and Sheth, P. Enhancing web services description and discovery to facilitate composition. in semantic web services and web process composition. In *Proceedings of the First International Workshop*, volume 3387 of Lecture Notes in Computer Science, San Diego, California, USA, 2004. Springer.

[22] Shen, Y. Y. Z. C. . W. C., J. From bpel4ws to owl-s: Integrating e-business process descriptions. In *Proceedings of 2nd IEEE international conference on services computing(SCC 2005)*, pages 181–188, Orlando, USA, July 2005.

[23] Thöne, D. R., S. and Engels, G. *Process-Oriented Flexible Composition of Web Services with UML*. LNCS Book chapter "Advanced conceptual modelling techniques", vol. 2784/2003, ISBN: 978-3-540-20255-4, 2003.

[24] Timm, J. and Gannod, G. A model-driven approach for specifying semantic web services. In *Proceedings of the 3rd IEEE International Conference on Web Services (ICWS 2005)*, pages 356–361, July 2005.