Semantic Annotations for WSDL and XML Schema

Editors' Copy $Date: 2007/06/12 18:00:53 $

This version:
http://www.w3.org/2002/ws/sawsdl/spec/
Latest version:
http://www.w3.org/TR/sawsdl/
Last Published version:
http://www.w3.org/TR/2006/WD-sawsdl-20060928/
Editors:
Joel Farrell, IBM
Holger Lausen, DERI Innsbruck

Abstract

This document defines a set of extension attributes for the Web Services Description Language and XML Schema definition language that allows description of additional semantics of WSDL components. The specification defines how ~~such~~ semantic annotation is accomplished using references to semantic models, e.g. ontologies. ~~SAWSDL~~ ^ does not specify a language for representing the semantic models. Instead it provides mechanisms by which concepts from the semantic models, typically defined outside the WSDL document, can be referenced from within WSDL and XML Schema components

using annotations.

Status of this Document

This section describes the status of this document at the time of its publication. Other documents may supersede this document.

This document is an editors' copy produced by the Semantic Annotations for WSDL Working Group that has no official standing.

Table Of Contents

Appendices

1. Introduction

Semantic Annotations for WSDL and XML Schema (SAWSDL) defines how to add semantic annotations to various parts of a WSDL document such as input and output message structures, interfaces and operations. The extension attributes defined in this specification fit within the WSDL 2.0 [WSDL 2.0], WSDL 1.1 [WSDL 1.1] and XML Schema [XMLSchema Part 1: Structures] extensibility frameworks. For example, this specification defines a way to annotate WSDL interfaces and operations with categorization information that can be used to publish a Web service in a registry. The annotations on schema types can be used during Web service discovery and composition. In addition, SAWSDL defines an annotation mechanism for specifying the data mapping of XML Schema types to and from an ontology; such mappings could be used during invocation, particularly when mediation is required. To accomplish semantic annotation, SAWSDL defines extension attributes that can be applied both to WSDL elements and to XML Schema elements.

The semantic annotations reference a concept in an ontology or a mapping document. The annotation mechanism is independent of the ontology expression language and this specification requires no particular ontology language. It is also independent of mapping languages and does not restrict the possible choices of such languages.

The rest of the document describes the SAWSDL extension attributes and provide examples as appropriate to illustrate their usage. For background and further examples, read the *Semantic Annotations for WSDL and XML Schema — Usage Guide* [SAWSDL Usage Guide]. Additionally, the SAWSDL Working Group plans to keep its Working Group page up to date (at least during the lifetime of the WG) with links to known uses of SAWSDL, including SAWSDL-aware tools, and most importantly, specifications (from W3C and ~~outside~~) that define concrete semantic models for SAWSDL annotations, and how

annotations with these models can be used.

1.1 Terminology

The following are the basic definitions for the terminology used in this document.

Semantic Model
: A semantic model is a set of machine-interpretable representations used to model an area of knowledge or some part of the world, including software. Examples of such models are ontologies that embody some community agreement, logic-based representations, etc. Depending upon the framework or language used for modelling, different terminologies exist for denoting the building blocks of semantic models.

Concept
: A concept is an element of a semantic model. This specification makes no assumptions about the nature of concepts, except that they must be identifiable by URIs. A concept can for example be a classifier in some language, a predicate logic relation, the value of the property of an ontology instance, some object instance or set of related instances, an axiom, etc.

Semantic Annotation
: A semantic annotation in a document is additional information that identifies or defines a concept in a semantic model in order to describe part of that document. In SAWSDL, semantic annotations are XML attributes added to a WSDL or associated XML Schema document, at the XML element they describe. Semantic annotations are of two kinds: explicit identifiers of concepts, or identifiers of mappings from WSDL to concepts or vice versa.

Semantics
: Semantics in the scope of this specification refers to sets of concepts identified by annotations.

1.2 Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.3 Namespaces

The XML namespace URIs [XML Namespaces] together with the prefixes as

used by this specification are as follows:

Prefix	Namespace name
sawsdl	http://www.w3.org/ns/sawsdl
sawsdlrdf	http://www.w3.org/ns/sawsdl#
wsdl	http://www.w3.org/ns/wsdl
wsdl11	http://schemas.xmlsoap.org/wsdl/
xs	http://www.w3.org/2001/XMLSchema

1.4 Example

In order to illustrate the concepts of SAWSDL, later sections will use a purchase order Web service interface. This imaginary Web service expects as input a customer account number and a list of items to be ordered, each containing quantity information and a product identifier in form of a Universal Product Code (UPC). The service will return the status of the order, which can be either reject, accept or pending. The WSDL including semantic annotations for this service is given below.

```
<wsdl:description
  targetNamespace="http://www.w3.org/2002/ws/sawsdl/spec/
wsdl/order#"
  xmlns="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#"
  xmlns:wsdl="http://www.w3.org/ns/wsdl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:sawsdl="http://www.w3.org/ns/sawsdl">

  <wsdl:types>
    <xs:schema targetNamespace="http://www.w3.org/2002/ws/
sawsdl/spec/wsdl/order#"
      elementFormDefault="qualified">
      <xs:element name="OrderRequest"
          sawsdl:modelReference="http://www.w3.org/2002/ws/
sawsdl/spec/ontology/purchaseorder#OrderRequest"
          sawsdl:loweringSchemaMapping="http://www.w3.
org/2002/ws/sawsdl/spec/mapping/RDFOnt2Request.xml">
        <xs:complexType>
```

```
          <xs:sequence>
            <xs:element name="customerNo" type="xs:
integer" />
            <xs:element name="orderItem" type="item"
minOccurs="1" maxOccurs="unbounded" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:complexType name="item">
        <xs:all>
          <xs:element name="UPC" type="xs:string" />
        </xs:all>
        <xs:attribute name="quantity" type="xs:integer" />
      </xs:complexType>
      <xs:element name="OrderResponse" type="confirmation" /
>
      <xs:simpleType name="confirmation"
          sawsdl:modelReference="http://www.w3.org/2002/ws/
sawsdl/spec/ontology/purchaseorder#OrderConfirmation">
        <xs:restriction base="xs:string">
          <xs:enumeration value="Confirmed" />
          <xs:enumeration value="Pending" />
          <xs:enumeration value="Rejected" />
        </xs:restriction>
      </xs:simpleType>
    </xs:schema>
  </wsdl:types>

  <wsdl:interface name="Order"
      sawsdl:modelReference="http://example.org/
categorization/products/electronics">
    <wsdl:operation name="order" pattern="http://www.w3.org/
ns/wsdl/in-out"
        sawsdl:modelReference="http://www.w3.org/2002/ws/
sawsdl/spec/ontology/purchaseorder#RequestPurchaseOrder">
      <wsdl:input element="OrderRequest" />
      <wsdl:output element="OrderResponse" />
    </wsdl:operation>
  </wsdl:interface>
</wsdl:description>
```

The annotations in this example appear as *modelReference* and *loweringSchemaMapping* attributes on schema and WSDL elements. Each

modelReference shown above identifies the concept in a semantic model that describes the element to which it is attached. For instance, the *OrderRequest* element is described by the "OrderRequest" concept in the ontology whose URI is "http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder." A *loweringSchemaMapping* is also attached to the *OrderRequest* element to point to a mapping, in this case an XML document, which shows how the elements within the *OrderRequest* can be mapped from semantic data in the model.

Sections of this example annotated WSDL document will be used in subsequent parts of the specification where these SAWSDL attributes are fully defined.

2. Annotation Mechanisms

This section introduces the two basic semantic annotation constructs that SAWSDL provides. Subsequent sections explain how they can be applied to WSDL and XML Schema documents.

Conceptually, WSDL 2.0 has the following components to represent service descriptions: Element Declaration, Type Definition, Interface, Binding and Service. Of these, the first three, namely Element Declaration, Type Definition and Interface deal with the abstract definition of a service while the latter two deal with service implementation. This specification focuses on semantically annotating the abstract definition of a service to enable dynamic discovery, composition and invocation of services. This specification does not address the annotation of service implementations. It provides generic reference mechanisms that can be applied, for example, to the aforementioned components. The references can point to concepts defined in semantic model or to mapping documents.

A summary of the extension attributes defined by SAWSDL is given below:

- an extension attribute, named **modelReference**, to specify the association between a WSDL or XML Schema component and a concept in some semantic model. It is used to to annotate XML Schema type definitions, element declarations, and attribute declarations as well as WSDL interfaces, operations, and faults.
- two extension attributes, named **liftingSchemaMapping** and **loweringSchemaMapping**, that are added to XML Schema element declarations and type definitions for specifying mappings between semantic data and XML.

SAWSDL allows multiple semantic annotations to be associated with WSDL

elements. Both schema mappings and a model references can contain multiple pointers. Multiple schema mappings are interpreted as alternatives whereas multiple model references all apply. SAWSDL does not specify any other relationship between them. More details about this are presented in the sections that follow.

2.1 SAWSDL Model Reference

This section defines the extension attribute *modelReference*. A model reference may be used with every element within WSDL and XML schema. However, SAWSDL defines its meaning only for wsdl:interface, wsdl:operation, wsdl:fault, xs:element, xs:complexType, xs:simpleType and xs:attribute.

The following XML schema excerpt defines the *modelReference* attribute.

```
…
<xs:attribute name="modelReference" type="listOfAnyURI" />
<xs:simpleType name="listOfAnyURI">
    <xs:list itemType="xs:anyURI"/>
</xs:simpleType>
…
```

The value of the *modelReference* attribute is a set of zero or more URIs, separated by whitespaces, that identify concepts in a semantic model. Each URI is a pointer to a concept in a semantic model and is intended to provide semantic information about the WSDL or XML Schema component being annotated.

The *modelReference* attribute allows multiple annotations to be associated with a given WSDL or XML Schema component via a set of URIs. These URIs may identify concepts expressed in different semantic representation languages. When a component is annotated with a *modelReference* that includes multiple URIs, each of them applies to the component, but no logical relationship between them is defined by this specification.

SAWSDL does not define any particular way to dereference model references, i. e., it does not prescribe how a client processor can obtain the document in which the semantic model is defined. It is recommended that the URI used for pointing to a semantic concept resolve to a document containing its definition. If the semantic model is expressed using XML, it could be placed directly within the WSDL document.

Sections 3.1 through 3.3 describe the use of *modelReference* for annotating WSDL documents, whereas Section 4.1 describes its use for annotating XML schema documents.

2.2 SAWSDL Schema Mapping

This section defines the extension attributes *liftingSchemaMapping* and *loweringSchemaMapping*. SAWSDL introduces schema mapping annotations to address post-discovery issues in using a Web service. Model references can be used to help determine if a service meets the requirements of a client, but there may still be mismatches between the semantic model and the structure of the inputs and outputs. For example, a client may have a given name and last name among its data, and these values need to be concatenated in the message to the Web service. A lowering schema mapping would take the client's semantic data and turn it into XML, and in the process it would perform the necessary concatenation to produce the full name. In general, lifting schema mappings *lift* data from XML to a semantic model, whereas lowering schema mappings *lower* data from a semantic model into an XML structure.

In a more complex scenario, a client may use WSDL with semantic annotations to describe the service it expects. Again, if the XML structures expected by the client and by the service differ, schema mappings can translate the XML structures into the semantic model where any mismatches can be understood and resolved.

Schema mapping relates the instance data defined by an XML Schema document with some semantic data defined by a semantic model. Such mappings are useful in general when the structure of the instance data does not correspond trivially to the organization of the semantic data. The mappings are used when mediation code is generated to support invocation of a Web service.

The following schema excerpt defines the *liftingSchemaMapping* and *loweringSchemaMapping* attributes.

```
…
<xs:attribute name="liftingSchemaMapping"
type="listOfAnyURI" />
<xs:attribute name="loweringSchemaMapping"
type="listOfAnyURI" />
<xs:simpleType name="listOfAnyURI">
    <xs:list itemType="xs:anyURI"/>
</xs:simpleType>
```

```
…
```

The usage of schema mapping attributes is further defined in Section 4.2.

2.3 Embedding Semantic Models

The semantic annotation mechanism defined by this specification does not rely on any particular semantic modeling language. It only requires that the semantic concepts defined in it be identifiable via URI references. The URIs typically refer to concepts in a semantic model that is external to the WSDL document. However, the URIs can also refer to elements within the WSDL document if semantic information is included in the document via WSDL extension elements as shown below on lines 3 to 14.

```
01   <wsdl:description … >
02
03     <rdf:RDF
04       xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"
05       xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
06       xmlns:owl="http://www.w3.org/2002/07/owl#"
07       xml:base="http://www.w3.org/2002/ws/sawsdl/spec/
ontology/purchaseorder">
08       <owl:Class rdf:ID="OrderRequest"/>
09       <owl:ObjectProperty rdf:ID="has_items">
10         <rdfs:domain rdf:resource="#OrderRequest"/>
11         <rdfs:range rdf:resource="#Item"/>
12       </owl:ObjectProperty>
13       <owl:Class rdf:ID="Item"/>
14     </rdf:RDF>
15
16     <wsdl:types>
17       <xs:element name="OrderRequest"
18           sawsdl:modelReference="http://www.w3.org/2002/
ws/sawsdl/spec/ontology/purchaseorder#OrderRequest">
19           …
20       </xs:element>
21       …
22     </wsdl:types>
23
24     <wsdl:interface name="Order">
25       …
```

```
26     </wsdl:interface>
27   </wsdl:description>
```

SAWSDL does not define an additional container element for embeding semantic models, since WSDL already allows extension elements within a *wsdl: description* element as illustrated above. In the example above the language expressing the semantic model uses the RDF ID mechanism to define the resources referenced, for instance line 8 defines "OrderRequest" in a semantic model. This identifier is then referenced using *modelReference* in line 18.

2.4 External Annotations (Non-Normative)

This specification defines annotations that are specified as attributes in the annotated WSDL and XML Schema documents. There are scenarios, however, which call for annotating documents externally. For example, a WSDL document can import an external WSDL or XML Schema and wish to add annotations to it. Adding the annotations inside the imported document may not be possible due to considerations like authority (the imported document comes from an external organization unwilling to add the annotations) or because the imported document is generated on the request from a dynamic source and the generating process does not allow adding annotations.

Without creating new syntactic constructs for annotating external WSDL and XML Schema documents, we can suggest two approaches in which our semantic annotations can be applied externally: XSLT pre-processing, and using RDF.

XSLT Pre-processing: as both WSDL and XML Schema are XML languages, it is natural to use XPath to identify the elements in the external document that should be annotated with SAWSDL (or other) annotations. Therefore, one can construct an XSLT stylesheet that mostly copies its input to its output, only in specific places it would add the appropriate annotation attributes. The output of the XSLT stylesheet would be an annotated version of the external document, ready for being processed with a SAWSDL-aware processor.

Using RDF: the WSDL specification [WSDL 2.0] contains Appendix C: IRI-References for WSDL 2.0 Components. This appendix defines unambiguous URIs for identifying WSDL components. The XML Schema Working Group is working on a component designators specification [XMLSchema: Component Designators] that will also provide URIs for XML Schema components. SAWSDL Section 6 presents RDF properties equivalent to the SAWSDL annotation

mechanisms, therefore one can have a WSDL document that imports an external WSDL or XML Schema document, and also contains an RDF graph embedded as an extension (see also Section 2.3 for an example of RDF embedded in a WSDL document), and the embedded RDF graph will provide all the annotations for the components imported from the external document.

There may be other mechanisms and we do not intend to mandate any particular way of capturing external annotations. Different mechanisms will have different properties, for instance XSLT pre-processing for adding annotations may be fragile in face of changing input (especially if relying on concrete element positions); this fragility may be mitigated somewhat in the RDF approach as it identifies the components, independent of where in the imported document they are defined.

3. Annotating WSDL Documents

In terms of the WSDL 2.0 component model, a model reference is a new property. In particular, when used on an element that represents a WSDL 2.0 Component (e.g. a *wsdl:interface* element representing an Interface component, a *wsdl:operation* element representing an Interface Operation component, a top-level *xs:element* representing an Element Declaration component, etc.), the *modelReference* extension attribute with a non-empty value introduces an OPTIONAL property {model reference} whose value is the set of URIs taken from the value of the attribute. An empty model reference or no model reference are both reflected by the absence of the {model reference} property on the given component.

The *modelReference* annotation on xs:element, xs:complexType, xs:simpleType and xs:attribute define the semantics of the input or output data of WSDL operations, as shown in Section 4. A *modelReference* on a WSDL operation or fault gives semantic information about that operation, while a *modelReference* on a WSDL interface provides a classification or other semantic descriptions of the interface.

3.1 Annotating Interfaces with Model Reference

This section defines how a *modelReference* can be used to annotate *interface* elements, to categorize them according to some model, to specify behavioral aspects or other semantic definitions. A *modelReference* on a WSDL *interface* element provides a reference to a concept or concepts in a semantic model that describe the Interface. The example below illustrates a categorization annotation

on an *interface* element.

```
…
<wsdl:interface name="Order" sawsdl:modelReference="http://
example.org/categorization/products/electronics">
  …
</wsdl:interface>
…
```

The *modelReference* in this example points to a category concept "electronics" in some semantic model. For taxonomies whose elements are not identifiable with a URI, the *modelReference* can point to a simple semantic model that contains the taxonomy reference information. This specification does not define such a categorization semantic model, but includes a non-normative example in Appendix C. SAWSDL does not constrain the form of the semantic model for categorization or that of any other semantic model specified in a *modelReference* on an interface.

When an interface extends one or more interfaces, the model references of the extended interfaces do not propagate to the new interface. In addition, annotations applied to other WSDL components, such as operations, are separate. SAWSDL defines no relationship between annotations.

In the WSDL component model, a non-empty modelReference on a WSDL interface is represented as {model reference} property of the Interface component; the case of an empty modelReference or no modelReference at all is represented with an Interface component that does not have a {model reference} property.

Appendix C further describes the use of categorization annotations including examples that show how to use the taxonomies supported by Universal Description, Discovery and Integration (UDDI) to categorize an interface.

3.2 Annotating Operations with Model Reference

The example below shows the annotated WSDL with a *modelReference* attribute to annotate the *operation* element. The annotation of the *operation* element carries a reference to a concept in a semantic model that provides a high level description of the operation, specifies its behavioral aspects or includes other semantic definitions. In the purchase order example, the *order* operation can be annotated using a class that represents a purchase order request operation. The annotation of the operation using the *modelReference* attribute is shown below.

```
…
<wsdl:operation name="order" pattern="http://www.w3.org/ns/
wsdl/in-out"
  sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/
spec/ontology/purchaseorder#RequestPurchaseOrder">
  <wsdl:input element="OrderRequest"/>
  <wsdl:output element="OrderResponse"/>
</wsdl:operation>
…
```

Although inputs and outputs provide one way of capturing the semantics of an operation, a simple semantic annotation indicating the intended behavior of a given operation as a verb concept may be useful at certain times. During service discovery, this verb provides a coarse indication of whether this service is a match for a given request. This can act as way to reduce the number of services whose input and output must be checked.

Operations can also be annotated with category references. These are separate from any categorizations applied to other WSDL components (e.g. the interface holding the operation) and SAWSDL defines no relationships between categorizations applied to them.

In the WSDL component model, a non-empty modelReference on a WSDL interface operation is represented as {model reference} property of the Interface Operation component; the case of an empty modelReference or no modelReference at all is represented with an Interface Operation component that does not have a {model reference} property.

3.3 Annotating Faults with Model Reference

The annotation of the *fault* element carries a reference to a concept in a semantic model that provides a high level description of the fault and can include other semantic definitions. The *fault* annotation does not describe the fault message, which should be annotated in the XML schema. The example below illustrates how to use the *modelReference* attribute to annotate the *fault* element. A fault that could be associated with *order* operation can be annotated using a class that represents the failure of a purchase order request operation due to the unavailability of the item. The annotation of a *fault* using the *modelReference* attribute is shown below.

```
…
```

```
<wsdl:interface name="Order">

  <wsdl:fault name="ItemUnavailableFault"
element="AvailabilityInformation"
    sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/
spec/ontology/purchaseorder#ItemUnavailable"/>
  …
</wsdl:interface>
```

This example identifies the "ItemUnavailable" concept in the referenced semantic model as a description of the fault "itemUnavailableFault."

In the WSDL component model a non-empty modelReference on a WSDL interface fault is represented as {model reference} property of the Interface Fault component; the case of an empty modelReference or no modelReference at all is represented with an Interface Fault component that does not have a {model reference} property.

4. Annotating XML Schema Documents

In order to annotate XML schema documents, SAWSDL provides two constructs: model reference and schema mapping. The former provides a generic link from an XML structure to a semantic model. XML Schema model reference annotations can be used, for example, to help determine if a service meets the requirements of a client. The schema mapping addresses post-discovery issues when using Web services, such as how to overcome structural mismatches between the semantic model and the service inputs and outputs.

4.1 Annotating XML Schema Documents with Model Reference

Model references define additional semantics for the annotated XML Schema components.

4.1.1 Annotating Simple Types with Model Reference

Simple types can be annotated by including a *modelReference* attribute on the *xs:simpleType* element. An example is shown below for the output of the *order* operation in the example WSDL document presented in Section 1.4.

```
…
<xs:simpleType name="Confirmation"
```

```
  sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/
spec/ontology/purchaseorder#OrderConfirmation">
  …
</xs:simpleType>
…
```

In this example, any element or attribute whose type is *Confirmation* is described by the *OrderConfirmation* concept in the referenced semantic model.

In the XML Schema component model, a non-empty modelReference on a top-level simple type is represented as {model reference} property of the XML Schema Simple Type Definition Schema component; the case of an empty modelReference or no modelReference at all is represented with an XML Schema Simple Type Definition component that does not have a {model reference} property. {model reference} properties are propagated from a simple type definition schema component to all attribute and element declaration schema components that are defined with that simple type.

In the WSDL component model, a non-empty modelReference on a top-level simple type used in WSDL is represented as {model reference} property of the WSDL Type Definition component; the case of an empty modelReference or no modelReference at all is represented with a Type Definition component that does not have a {model reference} property. {model reference} properties are propagated from a type definition WSDL component to all element declaration WSDL components that are defined with that type.

4.1.2 Annotating Complex Types with Model Reference

There are two principal techniques for annotating complex types which can be summarized as follows:

- Bottom Level Annotation: Annotating at the member element or attribute level
- Top Level Annotation: Annotating at complex type container level

In bottom level annotation, all the member elements and attributes in a complex type can be annotated. In some cases, the members of a complex type will correspond with the concepts in a semantic model. To accommodate this case, the member elements and attributes can be annotated by adding a modelReference attribute to the relevant schema element or attribute.

```
…
```

```
<xs:complexType>
  <xs:sequence minOccurs="1" maxOccurs="unbounded">
    <xs:element name="quantity" type="xs:integer"
      sawsdl:modelReference="http://www.w3.org/2002/ws/
sawsdl/spec/ontology/purchaseorder#Quantity"/>
    <xs:element name="UPC" type="xs:string"
      sawsdl:modelReference="http://www.w3.org/2002/ws/
sawsdl/spec/ontology/purchaseorder#ProductCode"/>
  </xs:sequence>
</xs:complexType>
…
```

In this *sequence*, each bottom level *element* has an annotation since the semantic model contains concepts, namely "Quantity" and "ProductCode", that describe each of the components of the complex type.

Bottom level annotation uses the mechanisms described below in sections 4.1.3 and 4.1.4.

In top level annotation, the complex types themselves are annotated with model references. If multiple concepts describe the complex type, all of their URIs can be included in the value of the *modelReference* attribute. This *sawsdl: modelReference* attribute on a complex type annotates the type as a whole, but does not necessarily make any specific statements about the elements or attributes within the complex type.

```
…
<xs:complexType sawsdl:modelReference="http://www.w3.
org/2002/ws/sawsdl/spec/ontology/
purchaseorder#OrderRequest">
  <xs:sequence minOccurs="1" maxOccurs="unbounded">
    <xs:element name="quantity" type="xs:integer"
    <xs:element name="UPC" type="xs:string"
  </xs:sequence>
</xs:complexType>
…
```

Here, the *complexType* as a whole has been annotated with a reference to the *OrderRequest* concept. *OrderRequest* describes a concept that groups "quantity" and "UPC" elements that make up the complex type.

A complex type can be annotated at both the top and bottom level. These

annotations are independent of each other.

In the XML schema component model, a non-empty modelReference on a top-level complex type is represented as {model reference} property of the XML Schema Complex Type Definition component; the case of an empty modelReference or no modelReference at all is represented with a type definition schema component that does not have a {model reference} property. {model reference} properties are propagated from a complex type definition schema component to all element declaration schema components that are defined with that complex type.

In the WSDL component model, a non-empty modelReference on a top-level complex type used in WSDL is represented as {model reference} property of the WSDL Type Definition component; the case of an empty modelReference or no modelReference at all is represented with a Type Definition component that does not have a {model reference} property. {model reference} properties are propagated from a type definition component to all element declaration components that are defined with that type.

4.1.3 Annotating Elements with Model Reference

An element declaration can be annotated by including a *modelReference* on the *xs:element* element. An example of annotating a global *element* using the *modelReference* attribute is shown below.

```
…
<xs:element name="OrderRequest"
    sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/
spec/ontology/purchaseorder#OrderRequest">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="customerNo" type="xs:integer" />
      <xs:element name="orderItem" type="item"
minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
…
```

In this example, the annotation indicates that the *element* "OrderRequest" is described by the concept "OrderRequest" in the referenced semantic model. This example is very similar to a top-level *complexType* annotation in that the

element being annotated is defined in terms of a *complexType* and the annotation describes the "OrderRequest" as a whole.

A non-empty modelReference on a top-level element declaration used in WSDL is represented as {model reference} property of the WSDL Element Declaration component or the XML Schema Element Declaration Component; the case of an empty modelReference or no modelReference ~~at all~~ is represented with a WSDL or XML Schema Element Declaration component that does not have a {model reference} property. Due to model reference propagation, element declaration {model reference} property also contains the values of the {model reference} property from the type definition component referenced by this element declaration.

4.1.4 Annotating Attributes with Model Reference

An attribute can be annotated by including a *modelReference* on the *xs:attribute* element. If the quantity element in the example above were defined as an attribute, a *modelReference* could be applied to it as follows.

```
…
<xs:attribute name="quantity" type="xs:integer"
    sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/
spec/ontology/purchaseorder#Quantity"/>
…
```

This annotation indicates that the *attribute* "quantity" is described by the concept "Quantity" in the referenced semantic model.

In the XML schema component model, a non-empty modelReference on a top-level attribute declaration is represented as {model reference} property of the XML Schema Attribute Declaration Schema component; the case of an empty modelReference or no modelReference at all is represented with an Attribute Declaration Schema component that does not have a {model reference} property. Due to model reference propagation, attribute declaration {model reference} property also contains the values of the {model reference} property from the simple type definition component referenced by this attribute declaration.

4.2 Annotating XML Schema Documents with Schema Mapping

The extension attributes *liftingSchemaMapping* and *loweringSchemaMapping* are used to associate a schema type or element with a mapping to an ontology.

Schema mappings may be added to global type definitions (complex or simple) as well as to global element declarations. It is possible to specify either lowering or lifting information as well as both together on the same schema element.

The value of the *liftingSchemaMapping* attribute is a set of zero or more URIs that reference mapping definitions. A mapping referenced by this attribute defines how an XML instance document conforming to the element or type defined in a schema is transformed to data that conforms to some semantic model, i.e. the output of the transformation process will be semantic data. The input to the transformation is the XML element on whose declaration the mapping is located; or an element valid according to the type on whose definition the mapping is located.

The value of the *loweringSchemaMapping* attribute is a set of zero or more URIs that reference mapping definitions. A mapping referenced by this attribute defines how data in a semantic model is transformed to XML instance data. The input will be some semantic data. The output of the process will be the XML element on whose declaration the mapping is located; or an element valid according to the type on whose definition the mapping is located.

When multiple URIs are specified on liftingSchemaMapping or loweringSchemaMapping, the schema mappings they reference are to be treated as alternatives, i.e. the client processor should choose one of them to apply, and the choice is fully at the client processor's discretion. For example, a mapping can be selected based on what mapping language the processor supports (different alternatives can use different languages), based on the availability of the mapping document, or by other preferences.

This specification provides a mechanism for associating optional schema mapping functions with a global type definition or global element declaration without any restriction on the choice of the mapping language. Just as SAWSDL does not prescribe any particular ontology representation language for specifying modelReferences, it does not prescribe any particular mapping representation language.

The following excerpt from the purchase order example shows how XSLT can be used as a schema mapping language to specify mapping from XSD elements to concepts in a semantic model. Detailed examples showing schema mapping using XSLT [XSLT] and SPARQL [SPARQL] are shown in Appendix A. Other languages, such as XQuery [XQuery], can also be used.

...

```
<xs:element name="OrderResponse" type="confirmation" />
<xs:simpleType name="confirmation"
    sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/
spec/ontology/purchaseorder#OrderConfirmation"
    sawsdl:liftingSchemaMapping="http://www.w3.org/2002/ws/
sawsdl/spec/mapping/Response2Ont.xslt">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Confirmed" />
    <xs:enumeration value="Pending" />
    <xs:enumeration value="Rejected" />
  </xs:restriction>
</xs:simpleType>
…
```

In order to perform the lifting of the data contained in an XML message, a client processor retrieves the transformation from the URI given in the value of sawsdl: liftingSchemaMapping (or from a cache or registry etc.) and applies it to those elements where the schema mapping was specified. In the example above *http://www.w3.org/2002/ws/sawsdl/spec/mapping/Response2Ont.xslt* will be applied to *OrderResponse* elements. A mapping specified on a global type definition can be applied to any element that is of that type.

A mapping specified on an element, including an empty value (""), overrides any mappings specified on its type. If no mapping is specified on the element, the mapping on its type applies to the element.

The following example illustrates this rule. The element *orderItem* has a *liftingSchemaMapping*. The complex type of *orderItem* is defined later in the schema and the complex type itself has a schema mapping. In such a case, the *liftingSchemaMapping* specified on the element overrides the one specified on the complex type. This means that only the schema mapping that is given with the URI http://example.org/mapping/OrderItem2Ont.xslt applies to the element orderItem. The mapping http://example.org/mapping/ItemType2Ont.xslt does not apply. The reason for specifying such an override rule is to allow an element to indicate the type mapping does not apply.

```
…
<xs:element name="orderItem" type="itemType"
  sawsdl:liftingSchemaMapping="http://example.org/mapping/
OrderItem2Ont.xslt"/>

<xs:complexType name="itemType"
  sawsdl:liftingSchemaMapping="http://example.org/mapping/
```

```
ItemType2Ont.xslt">
  <xs:sequence>
      <xs:element ref="partDesc" />
  </xs:sequence>
  <xs:attribute name="ItemID" type="xs:string"/>
</xs:complexType>
…
```

When used on an element that represents a WSDL 2.0 Component (a top-level *xs:element* element representing an Element Declaration component, an *xs:complexType* element or an *xs:simpleType* element representing a Type Definition component), the loweringSchemaMapping and liftingSchemaMapping extension attributes introduce the properties {lowering schema mapping} and {lifting schema mapping}.

When used on an element that represents an XML Schema Component (a top-level Element Declaration Schema component element, a Complex Type Definition Schema component or a Simple Type Definition Schema component), the loweringSchemaMapping and liftingSchemaMapping extension attributes introduce properties {lowering schema mapping} and {lifting schema mapping}.

The value of either of these properties is a (possibly empty) set of URIs taken from the value of the respective attribute. If an element declaration does not have a liftingSchemaMapping (or loweringSchemaMapping) attribute but its type definition does, the element declaration component acquires the {lifting schema mapping} (or {lowering schema mapping}) property from the type definition component.

5. WSDL 1.1 Support

The mechanism for semantic annotation described in this specification can also be applied to WSDL 1.1 [WSDL 1.1] Web service descriptions. All the XML attributes defined in this specification apply without modification to the WSDL 1.1 descriptions. However, in some cases they are applied to different elements in the WSDL document structure and a new element is introduced to facilitate operation annotations.

5.1 SAWSDL attrExtensions Element

The WSDL 1.1 schema does not allow extension attributes on the *operation* element, so this specification introduces a new extension element,

attrExtensions, to support semantic annotation of WSDL 1.1 operations. The following XML Schema excerpt defines the *attrExtensions* element.

```
…
<xs:element name="attrExtensions">
  <xs:complexType>
    <xs:anyAttribute namespace="##any"
processContents="lax" />
  </xs:complexType>
</xs:element>
…
```

The *attrExtensions* element provides a general mechanism for adding extension attributes where attribute extensibility is not allowed, but element extensibility is allowed. It SHOULD NOT be used where attribute extensibility is allowed. For SAWSDL it is used to add the sawsdl:modelReference attribute in WSDL 1.1 operations. It MUST NOT be used for SAWSDL annotations in WSDL 2.0. Attributes with the same namespace name and local name MUST NOT appear both on the attrExtensions element and on its parent element.

5.2 WSDL 1.1 Annotations

portTypes

A *portType* corresponds to a WSDL 2.0 *interface* and is annotated in the same way.

Input and Output

Annotation of XML Schema types with *modelReference*, *liftingSchemaMapping* or *loweringSchemaMapping* can be accomplished using the approach described for annotating these elements in WSDL 2.0. In addition, a *liftingSchemaMapping*, *loweringSchemaMapping* or *modelReference* attribute may be added to a *part* element (under a *message* element) to specify an input or output annotation that applies to the entire message part. Message parts are referenced from the *portType* structure in WSDL 1.1 that generally corresponds to the WSDL 2.0 *interface* structure.

Faults

In WSDL 1.1, faults are specified as messages within operations. In contrast to WSDL 2.0, a WSDL 1.1 fault is defined identically to an input or output, i.e. as a

fault subelement of the *operation* element. Annotation of the meaning of the fault needs to be done on the *fault* element in each operation where it occurs.

Operations

Because operation in WSDL 1.1 does not allow attribute extensibility, an *operation* is annotated by adding an *attrExtensions* element as a child of the *operation* element. The *modelReference* attribute of *attrExtensions* specifies the operation annotation. An example of such an annotation is shown below.

```
…
<wsdl11:operation name="order">
    <sawsdl:attrExtensions
        sawsdl:modelReference="http://www.w3.org/2002/ws/
sawsdl/spec/ontology/purchaseorder#RequestPurchaseOrder">
    <wsdl11:input message="OrderRequestMessage"/>
    <wsdl11:output message="OrderResponseMessage"/>
</wsdl11:operation>
…
```

5.3 Example in WSDL 1.1

The same example as shown in Section 1.4 is shown here in its WSDL 1.1 form. An important difference is the way messages are defined:

- The message structure is defined in terms of message parts. For the response message, the SAWSDL annotation is applied to the *part* element named *OrderResponse*. In the WSDL 2.0 example, this annotation was applied to the *OrderResponse* element in the schema. This annotation could have been left the same in the WSDL 1.1 example, but was applied in this way to illustrate the annotation of message parts.
- The definition of the *OrderResponse* element was not included in the schema since the *OrderResponse* message part fills its role.

The other difference is the inclusion of *attrExtensions* to annotate the operation.

```
<wsdl11:definitions
  targetNamespace="http://www.w3.org/2002/ws/sawsdl/spec/
wsdl/order#"
  xmlns="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#"
  xmlns:wsdl11="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
```

```
  xmlns:sawsdl="http://www.w3.org/ns/sawsdl"
  name="OrderService">

  <wsdl11:types>
    <xs:schema
      targetNamespace="http://www.w3.org/2002/ws/sawsdl/
spec/wsdl/order#"
      xmlns="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/
order#"
      elementFormDefault="qualified">
      <xs:element name="OrderRequest"
        sawsdl:modelReference="http://www.w3.org/2002/ws/
sawsdl/spec/ontology/purchaseorder#OrderRequest"
        sawsdl:liftingSchemaMapping="http://www.w3.org/2002/
ws/sawsdl/spec/mapping/Response2Ont.xslt">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="customerNo" type="xs:
integer" />
            <xs:element name="orderItem" type="item"
minOccurs="1" maxOccurs="unbounded" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:complexType name="item"
        sawsdl:modelReference="http://www.w3.org/2002/ws/
sawsdl/spec/ontology/purchaseorder#Item">
        <xs:all>
          <xs:element name="UPC" type="xs:string" />
        </xs:all>
        <xs:attribute name="quantity" type="xs:integer" />
      </xs:complexType>
      <xs:simpleType name="Confirmation">
        <xs:restriction base="xs:string">
          <xs:enumeration value="Confirmed"/>
          <xs:enumeration value="Pending"/>
          <xs:enumeration value="Rejected"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:schema>
  </wsdl11:types>

  <wsdl11:message name="OrderRequestMessage">
```

```
    <wsdl11:part name="OrderRequest"
element="OrderRequest" />
  </wsdl11:message>
  <wsdl11:message name="OrderResponseMessage">
    <wsdl11:part name="OrderResponse" type="Confirmation"
      sawsdl:modelReference="http://www.w3.org/2002/ws/
sawsdl/spec/ontology/purchaseorder#OrderConfirmation"/>
  </wsdl11:message>

  <wsdl11:portType name="Order">
    <wsdl11:operation name="order">
      <sawsdl:attrExtensions
        sawsdl:modelReference="http://www.w3.org/2002/ws/
sawsdl/spec/ontology/purchaseorder#RequestPurchaseOrder"/>
      <wsdl11:input message="OrderRequestMessage" />
      <wsdl11:output message="OrderResponseMessage" />
    </wsdl11:operation>
  </wsdl11:portType>
</wsdl11:definitions>
```

This example is available as a separate file at http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order11.

6. Mapping SAWSDL into RDF

This section describes how the WSDL extensions introduced in this document are mapped into an RDF form compatible with the WSDL 2.0 RDF Mapping [WSDL 2.0 RDF]. This specification introduces the properties {model reference}, {lifting schema mapping} and {lowering schema mapping}, and this section describes the equivalent RDF properties. However, since the WSDL 2.0 RDF mapping does not provide an RDF form for the element declaration and type definition components, the schema mapping properties (i.e. {lifting schema mapping} and {lowering schema mapping}) potentially present in SAWSDL documents, are currently not represented in the RDF form of WSDL documents. Only {model reference} properties are mapped.

The {model reference} property on any WSDL component (esp. Interface, Interface Operation, and Interface Fault components, as described earlier in this document) is represented in the RDF form using a property with the identifier *sawsdlrdf:modelReference* (the full URI of the property is then "http://www.w3.org/ns/sawsdl#modelReference"), as shown in table 6-1. As the value of {model reference} is a set of URIs, an RDF triple is introduced for each of the URIs.

Table 6-1. Mapping modelReference property to RDF

Property	RDF Form
	(componentId of the parent component generated per IRI-References for WSDL 2.0 Components)
{model reference}	*(for each URI <uri> in the value of {model reference})* <componentId> sawsdlrdf:modelReference <uri> .

Similarly, the {lifting schema mapping} and {lowering schema mapping} properties can be represented in RDF (within a hypothetical RDF mapping of XML Schema) using the properties sawsdlrdf:liftingSchemaMapping and sawsdlrdf:loweringSchemaMapping (the full URIs of the properties are then "http://www.w3.org/ns/sawsdl#liftingSchemaMapping" and "http://www.w3.org/ns/sawsdl#loweringSchemaMapping"). As the values of both {lifting schema mapping} and {lowering schema mapping} are sets of URIs, appropriate RDF triples are introduced for each of the URIs.

The following listing defines the three properties in RDF:

```
 <http://www.w3.org/ns/sawsdl#modelReference> rdf:type rdf:
Property .
 <http://www.w3.org/ns/sawsdl#liftingSchemaMapping> rdf:
type rdf:Property .
 <http://www.w3.org/ns/sawsdl#loweringSchemaMapping> rdf:
type rdf:Property .
```

7. References

7.1 Normative References

[RFC2119]
Key words for use in RFCs to Indicate Requirement Levels, S. Bradner, Author. Internet Engineering Task Force, June 1999. Available at http://www.ietf.org/rfc/rfc2119.txt.

[WSDL 1.1]
W3C Note, Web Services Description Language (WSDL) 1.1 Erik Christensen, Francisco Curbera, Greg Meredith and Sanjiva Weerawarana, Authors. Available at http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[WSDL 2.0]
Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language, R. Chinnici, J-J. Moreau, A. Ryman, S. Weerawarana, Editors. World Wide Web Consortium, 27 March 2006. This version of the "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language" Specification is available is available at http://www.w3.org/TR/2006/CR-wsdl20-20060327. The latest version of "Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language" is available at http://www.w3.org/TR/wsdl20.

[XML Namespaces]
Namespaces in XML, T. Bray, D. Hollander, and A. Layman, Editors. World Wide Web Consortium, 14 January 1999. This version of the Namespaces in XML Recommendation is http://www.w3.org/TR/1999/REC-xml-names-19990114. The latest version of Namespaces in XML is available at http://www.w3.org/TR/REC-xml-names.

[XMLSchema Part 1: Structures]
XML Schema Part 1: Structures, H. Thompson, D. Beech, M. Maloney, and N. Mendelsohn, Editors. World Wide Web Consortium, 28 October 2004. This version of the XML Schema Part 1 Recommendation is http://www.w3.org/TR/2004/REC-xmlschema-1-20041028. The latest version of XML Schema Part 1 is available at http://www.w3.org/TR/xmlschema-1.

7.2 Informative References

[GICS]
Global Industry Classification Standard. Available at http://www.msci.com/equity/gics.html.

[NAICS]
North American Industry Classification System. Available at http://www.census.gov/epcd/www/naics.html.

[OWL]
OWL Web Ontology Language Reference, Mike Dean and Guus Schreiber, Editors. W3C Recommendation, 10 February 2004, http://www.w3.org/TR/2004/REC-owl-ref-20040210/.
Latest version available at http://www.w3.org/TR/owl-ref/.

[SAWSDL Usage Guide]
Semantic Annotations for WSDL and XML Schema - Usage Guide, Rama Akkiraju and Brahmananda Sapkota, Editors. W3C Working Draft, 28 September 2006, http://www.w3.org/TR/2006/WD-sawsdl-guide-20060928/.
Latest version available at http://www.w3.org/TR/sawsdl-guide/.

[SPARQL]
SPARQL Query Language for RDF, Eric Prud'hommeaux and Andy Seaborne, Editors. World Wide Web Consortium, 6 April 2006. This version is http://www.w3.org/TR/2006/CR-rdf-sparql-query-20060406/. The latest version is available at http://www.w3.org/TR/rdf-sparql-query/.

[UDDI]
UDDI Version 3.0.2, Luc Clement, Andrew Hately, Claus von Riegen, and Tony Rogers, Editors. Organization for the Advancement of Structured Information Standards (OASIS). Available at http://uddi.org/pubs/uddi_v3.htm.

[UNSPSC]
The United Nations Standard Products and Services Code. United Nations Development Programme. Available at http://www.unspsc.org/.

[WSDL 2.0 RDF]
Web Services Description Language (WSDL) Version 2.0: RDF Mapping, J. Kopecky, B. Parsia, Editors. World Wide Web Consortium, 18 May 2006. This version of the "Web Services Description Language (WSDL) Version 2.0: RDF Mapping" Specification is available at http://www.w3.org/TR/2006/WD-wsdl20-rdf-20060518/. The latest version of "Web Services Description Language (WSDL) Version 2.0: RDF Mapping" is available at http://www.w3.org/TR/wsdl20-rdf/.

[XMLSchema: Component Designators]
XML Schema: Component Designator, M. Holstege and A.S. Vedamuth, Editors. World Wide Web Consortium, 29 March 2005. This version of the XML Schema: Component Designators Working Draft is http://www.w3.org/TR/2005/WD-xmlschema-ref-20050329/. The latest version of XML Schema: Component Designators is available at http://www.w3.org/TR/xmlschema-ref/.

[XQuery]
XQuery 1.0: An XML Query Language, Don Chamberlin , Anders Berglund, Scott Boag, *et. al.*, Editors. World Wide Web Consortium, 3 Nov 2005. This version is http://www.w3.org/TR/2005/CR-xquery-20051103/. The latest version is available at http://www.w3.org/TR/xquery/.

[XSLT]
XSL Transformations (XSLT) Version 2.0, Michael Kay, Editor. World Wide Web Consortium, 3 Nov 2005. This version is http://www.w3.org/TR/2005/CR-xslt20-20051103/. The latest version is available at http://www.w3.org/TR/xslt20/.

A. An Example (Non-Normative)

In this appendix we present the accompanying semantic descriptions and mappings to the example WSDL document presented in Section 1.4. We also provide examples on what the documents, pointed to by the schema mapping, might look like.

The files used in this example are:

- A WSDL file including SAWSDL annotation: http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order
- A purchase order ontology: http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder
- A lowering schema mapping using SPARQL and XSLT: http://www.w3.org/2002/ws/sawsdl/spec/mapping/RDFOnt2Request.xml
- A lifting schema mapping using XSLT: http://www.w3.org/2002/ws/sawsdl/spec/mapping/Response2Ont.xslt
- A lifting schema mapping using XQuery: http://www.w3.org/2002/ws/sawsdl/spec/mapping/Response2Ont.xq

The following subsections illustrate the two directions of the schema mapping: lifting and lowering. Lifting schema mapping transforms XML data into instances of a semantic model, and lowering schema mapping does the opposite, it transforms semantic model instances into XML data.

Since SAWSDL does not constrain the language of the semantic model nor the mapping language, the following can only be viewed as non-normative examples. In our setting we have chosen to base the examples on languages standardized by the W3C. For example, an RDF-based semantic Web service client may wish to invoke an operation that it earlier discovered. The client would have some RDF data that it intends to pass to the operation as the request XML message, lowering from RDF to XML. When the service replies with the response XML message, the client will need to lift the data back to RDF in order to process it. When using other languages to express the semantic model, mappings may be expressed using ~~different~~ languages.

A.1 Specifying Lowering Schema Mapping

When expressing the semantic model using RDF/XML, lowering schema mappings are potentially complex, because of the flexible nature of RDF/XML. RDF/XML is an XML serialization of the graph-of-triples model of RDF data. In essence, RDF data is always an unordered set of triples consisting of a subject, a predicate and an object. The XML serialization puts the values of these parts

of the triples alternatively into namespace-qualified XML element and attribute names, and into character values of elements or attributes. Languages like XSLT or XQuery rely on XPath to find and select data in XML structures. XPath is geared towards more fixed XML structures, so it takes different XPath expressions for instance to select predicates, which can be encoded as elements or attributes in many places within an RDF/XML document. XPath (and by extension XSLT and XQuery) does not provide means of ignoring the purely syntactical differences in the various possible RDF/XML serializations of otherwise equal RDF graphs, therefore all alternatives have to be accounted for in the XSLT stylesheet or XQuery query.

Practice has shown that it is a very hard task to create XSLT or XQuery transformations that take arbitrary RDF/XML as input. Instead, we can imagine that lowering schema mappings (for RDF-based semantic models) will use RDF query languages either to construct the final XML, or to construct a stricter form of XML that can then be input to XSLT or XQuery to produce the final XML structure. In our examples, we combine SPARQL to query the RDF data and produce an XML table of variable bindings, and XSLT that transforms this XML table into the required XML.

To conclude, if we use RDF as the base for our semantic model, we can easily use XML transformation technologies like XSLT or XQuery for lifting schema mappings, but for lowering schema mappings we use the XML technologies combined with an RDF query language like SPARQL to preprocess the RDF data.

In the following we present an XML file that captures the information necessary to perform the mapping from an RDF graph to XML data required by the Web service. The *lowering* element in this file contains two child elements. The first element *sparqlQuery* includes a SPARQL query to extract the data in form of a variable binding table; the second element is the root element of an XSLT transformation.

```
<lowering>
  <sparqlQuery>
    PREFIX po: <http://www.w3.org/2002/ws/sawsdl/spec/
ontology/purchaseorder#>
    SELECT ?qty ?UPC ?CustomerNo
    WHERE {
      ?order po:hasCustomer ?customer .
        ?customer po:hasCustomerID ?id .
          ?id po:hasLexicalRespresentation ?CustomerNo .
      ?order po:hasLineItems ?item .
```

```
        ?item po:hasQuantity ?qtyClass .
          ?qtyClass po:hasAmount ?qty .
        ?item po:hasProduct ?product .
          ?product po:hasProductCode ?code .
            ?code po:hasLexicalRespresentation ?UPC }
  </sparqlQuery>
  <xsl:transform version="2.0"
    xmlns:po="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/
order#"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:sp="http://www.w3.org/2005/sparql-results#">
    <xsl:output method="xml" version="1.0" encoding="iso-
8859-1" indent="yes" />
    <xsl:template match="/sp:sparql">
      <po:OrderRequest>
        <po:customerNo>
          <xsl:value-of
            select="sp:results/sp:result[position()=1]/sp:
binding[@name='CustomerNo']/sp:literal" />
        </po:customerNo>
        <xsl:apply-templates select="sp:results/sp:result" /
>
      </po:OrderRequest>
    </xsl:template>
    <xsl:template match="sp:result">
      <po:orderItem>
        <xsl:attribute name="quantity">
          <xsl:value-of select="sp:binding[@name='qty']/sp:
literal" />
        </xsl:attribute>
        <po:UPC>
          <xsl:value-of select="sp:binding[@name='UPC']/sp:
literal" />
        </po:UPC>
      </po:orderItem>
    </xsl:template>
  </xsl:transform>
</lowering>
```

We assume that some agent has semantic data as an RDF graph that is intended to be used for a Web service invocation. ~~Such data could look like in the listing below,~~ which shows an instance of the OrderRequest concept according to the purchase order ontology introduced earlier.

```
<!DOCTYPE rdf:RDF[
    <!ENTITY xs  "http://www.w3.org/2001/XMLSchema#" >
]> <rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns="http://www.w3.org/2002/ws/sawsdl/spec/ontology/
purchaseorder#"
  xml:base="http://www.w3.org/2002/ws/sawsdl/spec/ontology/
purchaseorder#">
  <owl:Ontology />
  <OrderRequest>
    <hasLineItems>
      <LineItem>
        <hasProduct>
          <Product>
            <hasProductCode>
              <UPCCode>
                <hasLexicalRespresentation>348290187318</
hasLexicalRespresentation>
              </UPCCode>
            </hasProductCode>
          </Product>
        </hasProduct>
        <hasQuantity rdf:datatype="&xs;float">12</
hasQuantity>
      </LineItem>
    </hasLineItems>
    <hasLineItems>
      <LineItem>
        <hasProduct>
          <Product>
            <hasProductCode>
              <UPCCode>
                <hasLexicalRespresentation>998212387318</
hasLexicalRespresentation>
              </UPCCode>
            </hasProductCode>
          </Product>
        </hasProduct>
        <hasQuantity>
          <Quantity>
            <hasAmount rdf:datatype="&xs;float">4</
```

```
hasAmount>
           <hasUnit rdf:resource="#Piece"/>
         </Quantity>
       </hasQuantity>
     </LineItem>
   </hasLineItems>
   <hasCustomer>
     <Customer>
       <hasCustomerID>
         <CustomerID>
           <hasLexicalRespresentation>007</
hasLexicalRespresentation>
         </CustomerID>
       </hasCustomerID>
     </Customer>
   </hasCustomer>
 </OrderRequest>
</rdf:RDF>
```

When the SPARQL is applied to the RDF graph above, the following XML will be generated as query answer.

```
<sparql
xmlns="http://www.w3.org/2005/sparql-results#">
  <head>
    <variable name="qty" />
    <variable name="UPC" />
    <variable name="CustomerNo" />
  </head>
  <results>
    <result>
      <binding name="qty">
        <literal>12</literal>
      </binding>
      <binding name="UPC">
        <literal>348290187318</literal>
      </binding>
      <binding name="CustomerNo">
        <literal>007</literal>
      </binding>
    </result>
    <result>
      <binding name="qty">
```

```
        <literal>4</literal>
      </binding>
      <binding name="UPC">
        <literal>998212387318</literal>
      </binding>
      <binding name="CustomerNo">
        <literal>007</literal>
      </binding>
    </result>
  </results>
</sparql>
```

When the XSLT is applied to the output of the SPARQL query, the following XML data is generated. This data corresponds to the element that has been annotated with the *loweringSchemaMapping* and can be used by some agent to invoke the purchase order Web service.

```
<ex:OrderRequest
xmlns:POOntology="http://example.org/ontologies/
purchaseorder#"
  xmlns:sp="http://www.w3.org/2005/sparql-results#"
  xmlns:po="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/
order#"
  xmlns:ex="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/
order#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2002/ws/sawsdl/spec/
wsdl/order# ../wsdl/SchemaOnly.xsd ">
  <ex:customerNo>007</ex:customerNo>
  <ex:orderItem quantity="12">
    <ex:UPC>348290187318</ex:UPC>
  </ex:orderItem>
  <ex:orderItem quantity="4">
    <ex:UPC>998212387318</ex:UPC>
  </ex:orderItem>
</ex:OrderRequest>
```

A.2 Specifying Lifting Schema Mapping

Our examples show lifting schema mappings using XSLT and XQuery, either one will take the XML data (e.g. coming from the Web service) and produce RDF

data in the RDF/XML syntax. This is rather straightforward. The mapping specification that is referenced using a *liftingSchemaMapping* takes as input XML data and produces semantic data as output. Within the purchase order example we have annotated the response type, i.e. *OrderResponse*. The XML data corresponding to the type can be transformed using XSLT or XQuery in order to obtain data processable by a semantic agent.

A possible response of the purchase order Web service is the following XML data.

```
<OrderResponse
  xmlns="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#"
  >Confirmed</OrderResponse>
```

The XSLT specification below defines how to convert the XML data to an RDF graph corresponding to the purchase ontology.

```
<xsl:transform version="2.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:po="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/
order#"
  xmlns:POOntology="http://www.w3.org/2002/ws/sawsdl/spec/
ontology/purchaseorder#">
  <xsl:output method="xml" version="1.0" encoding="iso-8859-
1" indent="yes" />
  <xsl:template match="/">
    <rdf:RDF>
      <POOntology:OrderConfirmation>
        <hasStatus rdf:datatype="http://www.w3.org/2001/
XMLSchema#string">
          <xsl:value-of select="po:OrderResponse" />
        </hasStatus>
      </POOntology:OrderConfirmation>
    </rdf:RDF>
  </xsl:template>
</xsl:transform>
```

When above XSLT is applied to the example response the following RDF instance will be produced as output.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
```

```
ns#"
         xmlns:po="http://www.w3.org/2002/ws/sawsdl/spec/
wsdl/order#"
         xmlns:POOntology="http://www.w3.org/2002/ws/sawsdl/
spec/ontology/purchaseorder#">
   <POOntology:OrderConfirmation>
      <hasStatus rdf:datatype="http://www.w3.org/2001/
XMLSchema#string">Confirmed</hasStatus>
   </POOntology:OrderConfirmation>
</rdf:RDF>
```

Equally ~~well one can specify the mapping~~ using XQuery. If both mappings are present, a processor can choose the one it is ~~able~~ to interpret.

```
xquery version "1.0";
declare namespace po="http://www.w3.org/2002/ws/sawsdl/spec/
wsdl/order#" ;
declare namespace rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#";
declare namespace owl="http://www.w3.org/2002/07/owl#" ;
<rdf:RDF>
  <po:OrderConfirmation>
    <po:hasStatus rdf:datatype="http://www.w3.org/2001/
XMLSchema#string">{ fn:string(/) }</po:hasStatus>
  </po:OrderConfirmation>
</rdf:RDF>
```

B. Categorization Examples (Non-Normative)

One purpose for annotating services at the interface level is to ~~help~~ enable dynamic discovery. This is possible when services are published, catalogued and annotated with semantics. The *modelReference* mechanism adds categorization information to interfaces which could be used while publishing services in registries such as UDDI [UDDI]. Users can choose any categorization of their choice, such as NAICS [NAICS], UNSPSC [UNSPSC] and GICS [GICS]. This aids in service discovery by narrowing the range of candidate services. The categorization can be used as input when the service is published in a UDDI registry since taxonomies supported by UDDI can be specified via a semantic model.

Service categorization is also aimed at supporting specialized taxonomies of middleware or utility services such as mediators. The categorization

modelReference is used to ensure that there is basic and high-level categorization information about a service.

In this example, the interface categorization *modelReference* from Section 3.1 points to an RDF instance that gives two pieces of information: the taxonomy value (category code) of the category and the URI of the taxonomy.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"
     xmlns:if="http://www.w3.org/2002/ws/sawsdl/spec/
ontology/interface#">
  <if:Category rdf:about="http://example.org/categorization/
products/electronics"
    if:hasValue="443112"
    if:usesTaxonomy="http://naics.com/"/>
</rdf:RDF>
```

This instance data derives from the following simple example model.

```
<rdf:RDF>
  <rdfs:Class rdf:about="http://www.w3.org/2002/ws/sawsdl/
spec/ontology/interface#Category"/>
  <rdf:Property rdf:about="http://www.w3.org/2002/ws/sawsdl/
spec/ontology/interface#hasValue">
    <rdfs:domain rdf:resource="http://www.w3.org/2002/ws/
sawsdl/spec/ontology/interface#Category"/>
    <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Literal"/>
  </rdf:Property>
  <rdf:Property rdf:about="http://www.w3.org/2002/ws/sawsdl/
spec/ontology/interface#usesTaxonomy">
    <rdfs:domain rdf:resource="http://www.w3.org/2002/ws/
sawsdl/spec/ontology/interface#Category"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/
XMLSchema#anyUri"/>
  </rdf:Property>
</rdf:RDF>
```

Using this simple model, it is possible to create categorizations using the registered taxonomies in UDDI. This information can then be reused when the service is published in the UDDI registry. In the following example we use the NAICS taxonomy identifier registered in UDDI, as opposed to identifying the

taxonomy with the URI of the NAICS Association Web site, as shown above.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"
     xmlns:if="http://www.w3.org/2002/ws/sawsdl/spec/
ontology/interface#">
  <if:Category rdf:about="http://example.org/categorization/
products/electronics"
    if:hasValue="443112"
    if:usesTaxonomy="uddi:uddi.org:ubr:categorization:
naics:2002"/>
</rdf:RDF>
```

When publishing in UDDI V3, the category bag created to classify the service would be built from this information. This is the UDDI *categoryBag* structure that would derive from this categorization.

```
<categoryBag>
         <keyedReference
         tModelKey="uddi:uddi.org:ubr:categorization:
naics:2002"
         keyName="Electronics"
         keyValue="443112"/>
</categoryBag>
```

The other UDDI taxonomies would also be referenced via their registered URIs.

C. XML Schema for Semantic Annotations for WSDL and XML Schema

The SAWSDL schema is also available as a separate file at sawsdl.xsd.

```
<xs:schema
  targetNamespace="http://www.w3.org/ns/sawsdl"
  xmlns="http://www.w3.org/ns/sawsdl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:wsdl="http://www.w3.org/ns/wsdl">

  <xs:simpleType name="listOfAnyURI">
    <xs:list itemType="xs:anyURI"/>
  </xs:simpleType>
```

```
  <xs:attribute name="modelReference" type="listOfAnyURI" />
  <xs:attribute name="liftingSchemaMapping"
type="listOfAnyURI" />
  <xs:attribute name="loweringSchemaMapping"
type="listOfAnyURI" />

  <xs:element name="attrExtensions">
    <xs:complexType>
      <xs:annotation>
        <xs:documentation>This element is for use in WSDL
1.1 only. It does not apply to WSDL 2.0 documents.  Use in
        WSDL 2.0 documents is invalid.</xs:documentation>
      </xs:annotation>
      <xs:anyAttribute namespace="##any"
processContents="lax" />
    </xs:complexType>
  </xs:element>
</xs:schema>
```

D. Acknowledgements (Non-Normative)

This document is the work of the W3C Semantic Annotations for Web Service Description Language Working Group.

Members of the Working Group are (at the time of writing, and in alphabetical order): Rama Akkiraju (IBM Corporation), Carine Bournez (W3C), J.B. Domingue (The Open University), Joel Farrell (IBM Corporation), Laura Ferrari (Telecom Italia SpA), Laurent Henocque (ILOG, S.A.), Mathias Kleiner (ILOG, S.A.), Jacek Kopecký (DERI Innsbruck at the Leopold-Franzens-Universität Innsbruck, Austria), Holger Lausen (DERI Innsbruck at the Leopold-Franzens-Universität Innsbruck, Austria), Peter Matthews (CA), Antony Miguel (Scapa Technologies Limited), John Miller (University of Georgia Research Foundation, Inc (UGARF)) Carlos Pedrinaci (The Open University), Eric Prud'hommeaux (W3C), Brahmananda Sapkota (DERI Galway at the National University of Ireland, Galway, Ireland), Amit Sheth (Wright State University), Claudio Venezia (Telecom Italia SpA), Tomas Vitvar (DERI Galway at the National University of Ireland, Galway, Ireland).

E. Change Log (Non-Normative)

Table E-1. Summary of Semantic Annotations for WSDL and XML Schema Specification Changes

Date	Author	Description
20070612	JAF	Corrected attrExtensions examples.
20070528	HL	Editorial comments Jack + fixed lifting schema mapping example
20070403	JAF	Using WSDL namespace from the new LC spec from the WSDL WG.
20070315	JAF	Changed namespace - CR issue 9 and propagation of model reference to address CR issue 7.
20070312	JAF	Removed interface model reference inheritance and documented Schema component model propagation of model reference to address CR issues 5, 6 and 8 and included references to the XML Schema component model called for by CR issue 1.
20070205	JAF	Addressed John Miller's editorial comments and refected the names space change made in the CR publication version.
20070201	JAF	Addressed CR issue 4
20061213	HL, JK	implementing LC issues 11, 13, 17
20061211	JAF	Final changes for the reorganization of the document structure to address LC issue 3
20061208	HL	Own section for embedding semantic descriptions, addressing issue 6
20061208	HL	Incorporated explanation suggested by Jaceck on issue 5
20061208	HL	Changed title to include " and XML Schema" assuming that working group title remains unchanged, but usage guide also adopts new title.
20061208	HL	Updated purchase order ontology to include concept Quantity, adopted lowering example accordingly and changed #ItemCode to #ProductCode to macht Ontology
20061208	HL	Change of structure to reflect the difference between WSDL and XSD annotation.

20061102	JAF	Resolution to LC Issue 1 on WSDL Component model throughout section 2 - editorial changes for clarity. Resolution to LC Issue 2 making clear when attrExtensions can be used. Editorial comments from Ramkumar Menon.
20061024	JAF	Editorial changes: Added examples to 2.1.5, replaced example in 2.1.6, corrected numbering of 2.1.7, added missing bold formatting from some examples in 2.1., added explanation to fault annotation to remind reader that the fault message is annotated separately.
20062009	HL	Corrected links, sorted references, removed redundant namesapaces from example.
20061909	JAF	Resolutions from 18 Sept. WG call - Section 2 text, Terminology, UDDI example, and references.
20060709	JAF	Clean up for Last Call publication.
20060609	JAF	Included all major editorial decisions from f2f Aug 30, 31.
20063108	HL	Included RDF Mapping
20062308	HL	Implemented editorial comments of Jack, added example for inline reference of semantic model.
20062108	JAF	Documented resolutions to issues 26 and 27.
20061508	JAF	Documented remainder of the resolution to issue 22 on WSDL 1.1 (operation annotation).
20062107	JAF	Documented resolution to issue 22 on WSDL 1.1 (but operation annotation question still open) and issue 23 on interface extensions. Also updated semantics definition based on 18 July 2006 WG call.
20061007	HL	Replaced XSLT example with "more correct" version.
20061007	HL	Removed relativ looking URIs from modelReference examples and replaced with full URIs.
20060628	JAF	Made all textual changes required for publication as 1st Public Draft, as indentified in 27 June 2006 WG call.
20060626	JAF	Completed documentation of decisions from f2f in Galway, including resolutions to issues 6, 7, 8, 9 and 14 on schema mapping, and 10 on categorization as a modelReference. (Some done in HL's previous update)
20060620	HL	Reordered Appendixes according to f2f. Referenced copy of RNet ontology. changed text on semantic models

20060614	JAF	Documented resolution to Issue 18: Relationship between modelReferences. Documented initial part of Issue 6 on lifting and lowering mappings. Continued with section 2 restructuring.
20060607	JAF	Documented resolution to Issue 5: Multiplicity of modeReferences. Documented initial part of Issue 6: Clarification of SchemaMapping concept.
20060530	JAF	Documented resolution to Issue 2: embedding semantic descriptions. Removed some WSDL 1.1 terminology.
20060523	HL	changed categoryname from xs:NcName to xs:string.
20060523	HL	Introduced simple example in Section 1, modified structure 2 to reflect the elements defined. (Addressed Issue 1 and 9).
20060519	JAF	Split References into Normative and Informative and brought them up to TR quality.
20061505	HL	made sawsdl scheme independent of wsdl xsd by resolution in http://www.w3.org/2002/ws/sawsdl/minutes/20060509
20060508	HL	Created sawsdl xsd, removed example.org imports for sawsdl schema.
20060509	HL	Created separate files for documents in purchase order example (grouped xsds to one file). Corrected several errors (mainly namespace), adopted xslt and XQuery examples respectively.
20060504	JAF	Based on 02 May 06 call - added requirements on semantic models, example.org for examples namespace, added todo list. More editorial clean up.
20060418	JAF	Changed name of spec based on decision in 18 April 06 call. Also editorial clean up.
20060414	JAF	Removed non-spec material from Mapping Choices Appendix.