Semantic Annotations for WSDL

Editors' Copy $Date: 2006/08/23 19:52:38 $

This version:
http://www.w3.org/2002/ws/sawsdl/spec/
Latest version:
http://www.w3.org/2002/ws/sawsdl/spec/
Editors:
Joel Farrell, IBM
Holger Lausen, DERI Innsbruck

Abstract

Semantics play an important role in (semi) automating the discovery of Web services, which is increasingly becoming a requirement in many domains including business process integration. The syntactic description of Web services written in Web Services Description Language (WSDL) format is not amenable to enabling such automation. Semantic Annotations for WSDL (SAWSDL) specification defines mechanisms using which semantic annotations can be added to WSDL components. These semantics when expressed in formal languages can help disambiguate the description of Web services. This makes it possible to automate Web service matching, composition and invocation.

SAWSDL does not specify a language for representing the semantic models e.g. ontologies. Instead it provides mechanisms by which concepts from the semantic models that are defined either within or outside the WSDL document can be referenced from within WSDL components as 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 Web Services Description Language Working Group that has no official standing.

Deleted: This document describes a set of extension attributes for the Web Services Description Language Version 2.0 [WSDL 2.0] that describes additional semantics of WSDL 2.0 components. The specification defines how such semantic annotation is accomplished using references to semantic models, e.g. ontologies, defined outside the WSDL 2.0 document.¶

Table of Contents

Deleted: a

1. Introduction (Informative)

Formatted: Left

As the number of available Web Services on the Internet and in within companies expands, it becomes increasingly important to have automated tools to help identify the Web services that match a requester's requirements. Adding semantics to represent the requirements and capabilities of Web services in unambiguous and preferably machine interpretable languages is essential for achieving this automation during discovery and invocation. For example, during development, a service provider can explicate the intended semantics by annotating the appropriate parts of the Web service with concepts from a richer semantic model. Since semantic models provide agreement on semantics of terms, and may provide formal and informal definitions of the entities, there will be less ambiguity in the intended semantics of the provider. During discovery, a service requestor can describe the service requirements using terms from the semantic model. Reasoning techniques can be used to find the semantic similarity between the service

description and the request. During composition, these annotations can be used to aggregate the functionality of multiple services to create useful service compositions. Finally during invocation, mappings to the semantic models can be used for data transformations. Therefore, once represented, semantics can be leveraged by tools to automate service discovery, mediation, composition and monitoring.

While Web Services Description Language (WSDL) provides a standard way to describe the interfaces of a Web Service at a syntactic level and how to invoke it, it does not explicitly define mechanisms to add semantic annotations to WSDL components. Semantic Annotations for WSDL (SAWSDL) specification defines such mechanisms using which semantic annotations can be added to WSDL components. These semantics when expressed in formal languages (such as OWL [OWL]) can help disambiguate the description of Web services. This makes it possible to automate Web service matching, composition and invocation.

Specifically, SAWSDL defines extension attributes that can be applied to both WSDL elements and XML Schema elements to annotate various parts of WSDL document such as input and output messages, the interface, the operation, the simple types, and the complex types. For example, it 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 inputs, outputs and schema types can be used during Web service matching and composition. In addition, SAWSDL defines an annotation mechanism for specifying the structural mapping of XML Schema types to and from an ontology which could used during invocation. Semantic annotations are references from an element within a WSDL or XML Schema document to a concept in a semantic model (alternatively an ontology) that is defined and maintained outside the WSDL document. SAWSDL does not prescribe any particular semantic representation language for representing these semantic models, or suggest hosting them directly in WSDL documents.

The extension attributes defined in this specification fit within the WSDL2.0 extensibility framework. The rest of the document describes the schema for the SAWSDL extension attributes and provide examples as appropriate to illustrate their usage.

This document is designed to be read by those interested in the technical details of SAWSDL. It is not particularly intended for the casual reader. The casual reader and the developers of semantic matching engines are suggested to read the SAWSDL Guide first.

Deleted: The specification of Semantic Annotations in WSDL (SAWSDL) defines how to add semantic annotations to WSDL 2.0 components.

Deleted: The specification

Deleted: S

Deleted: for example

Deleted: defined in a WSDL 2.0 interface.¶

Deleted: S

Deleted: .

Deleted: This specification defines annotation mechanisms for relating WSDL inputs and outputs to concepts defined in an outside ontology. Similarly, it defines how to annotate WSDL operations and how to categorize WSDL interfaces. Further, it defines an annotation mechanism for specifying the structural mapping of XML Schema types to and from an ontology. The annotation mechanism is independent of the ontology expression language and this specification requires no particular ontology language.

1.1 Terminology

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

Semantics

Semantics in the scope of this specification refers to concepts in a domain and the relationships between those concepts.

Semantic Model

A semantic model usually includes concepts in the domain of interest, relationships among them, their properties, and their values. Usually this is described as an ontology that embodies some community agreement.

Deleted: A semantic model captures the terms and concepts used to describe and represent an area of knowledge or some part of the world, including a software system.

Semantic Annotation

A semantic annotation is additional information in a document that identifies or defines the semantics of a part of that document. In SAWSDL, the semantic annotations are XML attributes added to a WSDL or associated XML Schema document. They establish the context of elements in WSDL document by referring to a part of a semantic model.

Deleted: meaning

Concept

A concept is a semantic definition. When referring to a concept within a semantic model, this specification does not make an implicit statement about its type. A concept within a semantic model might be an instance, class or property.

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 RFC2119 [RFC2119].

1.3 Namespaces

The XML namespace [XML Namespaces] URIs [URI] together with the prefixes as used by this specification are as follows:

Prefix	Namespace name
sawsdl	http://www.w3.org/2002/ws/sawsdl/spec/sawsdl#
wsdl	http://www.w3.org/2006/01/wsdl
xs	http://www.w3.org/2001/XMLSchema

1.4 Running 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 an Universal Product Code (UPC). The service will return the status of the order, which can be either reject, accept or pending. The WSDL for this service is given below. Future sections illustrate how to add semantic annnotations.

Deleted: European Article Number (EAN).

Deleted: Later examples will

```
<?xml version="1.0" encoding="UTF-8"?>
<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/2006/01/wsdl"
```

```
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:sawsdl="http://www.w3.org/2002/ws/sawsdl/spec/sawsdl#">

  <wsdl:types>
    <xs:schema
targetNamespace="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#"
elementFormDefault="qualified">
       <xs:element name="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>
      <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">
        <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">
     <wsdl:operation name="order"
pattern="http://www.w3.org/2006/01/wsdl/in-out">
        <wsdl:input element="OrderRequest"/>
        <wsdl:output element="OrderResponse"/>
     </wsdl:operation>
  </wsdl:interface>
</wsdl:description>
```

Deleted: EAN

The next section defines the extension attributes and the mechanisms using which semantic annotations can be added to WSDL components.

Deleted: ¶

2. Annotation Mechanism

Deleted: This section describes how semantic annotations are added to WSDL components.¶

Conceptually, WSDL 2.0 has the following components to represent service descriptions: types, interface, operation, binding, service and endpoint. Of these, the first three, namely types, interface and operation, deal with the abstract definition of a service while the remaining three 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 URI reference mechanisms via extensibility attributes. These are applied to WSDL type, interface and operation components to point to the semantic constructs defined in domain models.

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

- an extension attribute, named **modelReference**, to specify the association between a WSDL component and a concept in some semantic model. It is used to to annotate XSD complex type definition, simple type definition, element declaration, and attribute declaration as well as WSDL operations and interfaces.
- two extension attributes, namely **liftingSchemaMapping** and **loweringSchemaMapping**, that are added to XML Schema element declaration, complex type definitions and simple type definition for specifying mappings between semantic data and XML. These schema mappings can be used during invocation.

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.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:description … >

  <rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"

xml:base="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#
">
    <owl:Class rdf:ID="OrderRequest"/>
    <owl:ObjectProperty rdf:ID="has_items">
      <rdfs:domain rdf:resource="OrderRequest"/>
      <rdfs:range rdf:resource="Item"/>
    </owl:ObjectProperty>
    <owl:Class rdf:ID="Item"/>
  </rdf:RDF>

  <wsdl:types>
    …
  </wsdl:types>

  <wsdl:interface name="Order">
    …
  </wsdl:interface>
</wsdl:description>
```

Deleted: bility

Deleted: necessary for communication.

Deleted: document

Deleted: me

SAWSDL allows multiple semantic annotations to be associated with WSDL elements. Both a schema mapping and a model reference can contain a list of pointers. The schema mappings are interpreted as alternatives whereas the model references all apply. SAWSDL does not specify any other relationship between them. More details about this are presented in the sections to follow.

2.1 SAWSDL Model Reference

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

Deleted: A

Deleted: , h

Deleted: owever

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

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

The value of a *modelReference* attribute is a list of one or more URIs, separated by whitespaces, that reference concepts in a semantic model. Each URI is a pointer to a concept in a semantic model and is intended to provide additional semantic information about the component of the WSDL document that is being annotated. Sections 2.1.1 through 2.1.6 provide examples for using modelReferences on interfaces, operations, simple types, complex types, elements and attributes respectively.

Formatted: Font: Italic

Deleted: the

Deleted: referenced

Deleted: describes the annotated

SAWSDL allows multiple annotations to be associated with WSDL components. These annotations could be pointing to concepts from the same model or from a different model. If latter, these models could be defined using the same semantic representation language or a different one. For SAWSDL they are all URIs. When a WSDL component is annotated with a *modelReference* that includes multiple URI references to the model, each of them applies to the component, but no logical relationship between them is defined by SAWSDL.

Deleted: Model references define the additional semantics for the components of the WSDL instance document.¶

Deleted: our specification.

When used on an element that represents a WSDL 2.0 Component (e.g. wsdl:interface, wsdl:operation, top-level xsd:element, etc.), the *modelReference* extension attribute introduces an OPTIONAL property {model reference} whose value is a list of URIs taken from the value of the attribute. The absence of the {model reference} property is equal to its presence with an empty value.

Comment [I1]: <Rama> I don't understand this paragraph. What does { model reference} mean? How is it different from *modelReference*? </Rama>

SAWSDL does not define a particular way for dereferencing 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

resolves to a document containing its definition. If the semantic model is expressed using XML it could be placed directly within the WSDL document.

The *modelReference* annotations on xs:element, xs:complexType, xs:simpleType and xs:attribute define the semantics of the input or output of WSDL. A *modelReference* on a WSDL operation gives semantic information about that operation. One can use a *modelReference* on an interface and operation to associate preconditions and effects. A *modelReference* on a WSDL interface could be used as a service registry categorization annotation via referencing a classification scheme. The following subsections provide additional details for each one of these.

2.1.1 Annotating Interfaces With Model Reference

This section defines how a *modelReference* can be used to annotate interfaces, including how to categorize them according to some model. A *modelReference* on a WSDL *interface* component provides a reference to a concept or concepts in a semantic model that describe the interface. This specification describes specifically how to use *modelReference* to categorize the interface according to some taxonomy. The example below illustrates a categorization annotation on an interface.

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

The *modelReference* in the above examples points to a category in the semantic model called ‘categorizationSchemeA’. For taxonomies that are not expressed as part of a semantic model, 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 D. 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* is defined by extending one or more interfaces, the model references of the extended interfaces all apply to the new interface. In this way, an interface can be categorized via the modelReferences of the interfaces it extends. Categorizations applied to other WSDL components, such as operations, are separate. SAWSDL defines no relationship between categorizations applied to other WSDL components.

A *modelReference* on an interface can also be used to specify preconditions and effects that are applicable at the level of that interface. Examples for using a *modelReference* to define preconditions and effects is given in the SAWSDL Guide [SAWDL Guide].

2.1.2 Annotating Operations With Model Reference

Deleted: alternatively

Deleted: an

Deleted: also

Deleted: operations that refer to these schema elements

Formatted: Font: Italic

Deleted: ,

Deleted: while a

Deleted: provides e.g. a

Deleted: classification of the interface via references to

Deleted: a

Deleted: One purpose for annotating services at the interface level is to help enable dynamic discovery. This is possible when services are published, cataloged and annotated with semantics. The *modelReference* mechanism adds categorization information to services which could be used while publishing services in registries such as Universal Description, Discovery and Integration (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. Additionally, it can constitute the effective categorization when the service is made available via Web Services Inspection Language [WSIL] or some other solution-specific means. Service categorization is also aimed at supporting specialized taxonomies of middleware or utility services such as mediators. The category *modelReference* is used to ensure that there is basic and high-level categorization information about a service.¶

Formatted: Font: Italic

Deleted: extension

Deleted: of

Deleted: a

Formatted: Font: Italic

Formatted: Font: Italic

The example below illustrates how to use the *modelReference* attribute to annotate the *operation* element in the example introduced in Section 1.4. The annotation of the *operation* element carries a reference to a concept in a semantic model which provides a high level description of the operation. 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/2006/01/wsdl/in-out"

sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/p
urchaseorder#RequestPurchaseOrder">
  <wsdl:input element="OrderRequest"/>
  <wsdl:output element="OrderResponse"/>
</wsdl:operation>
…
```

Although inputs and outputs provide a logical 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. That way, more detailed matching of inputs and outputs can be deferred until a reasonable match at the level of an operation is obtained thereby saving time during the matching of a given request with a large number of services.

Operations can also be annotated with category references. These are separate from any categorizations applied to other WSDL components. SAWSDL defines no relationships between categorizations applied to them.

Just as a *modelReference* on an interface can be used to associate preconditions and effects that apply at the interface level, a *modelReference* on an operation can be used to associate preconditions and effects that apply at an operation level. Examples for illustrating such usage of *modelReference* is provided in SAWSDL Guide [SAWDL Guide].

Deleted: Semantic models, including categorization taxonomies, may themselves specify propagation of their annotations from interface to operations, in which case any needed precedences rules would be defined by the model.

Deleted: and

Formatted: Font: Italic

Formatted: Font: Italic

Formatted: Font: Italic

2.1.3 Annotating Simple Types With Model Reference

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

Deleted: To annotate s

Deleted: ,

Deleted: e

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

sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/o
rder#OrderConfirmation">
  …
```

```
</xs:simpleType>
…
```

more text about meaning

2.1.4 Annotating Complex Types With Model Reference

Two of the principal techniques for annotating complex types can be summarized as follows:

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

In the bottom level annotation approach, 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 definitions. Bottom level annotation is formally defined under Annotating Elements With Model Reference and Annotating Attributes With Model Reference.

In the top level annotation approach, complex types themselves are annotated with a *modelReference*. 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.

A complex type can be annotated at both the top and member level. Both the high-level type annotation and the specific member element or attribute annotation apply to the member.

Deleted: possible,

Comment [12]: <Rama> I don't understand this statement </Rama>

Formatted: Font: Italic

Deleted: *the*

Formatted: Font: Italic

Deleted: *r*

Comment [13]: <Rama> This sentence contradicts the sentence made at the end of the previous paragraph </Rama>

2.1.5 Annotating Elements With Model Reference

An element can be annotated by including a *modelReference* on the *xs:element* element. An example of annotating the member elements of a complex type with the *modelReference* attribute is shown below.

Deleted: To annotate

Deleted: s

Deleted: ,

Deleted: e

```
…
<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/p
urchaseorder#Quantity"/>
    <xs:element name="ISBN" type="xs:string"

sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/p
urchaseorder#ItemCode"/>
```

```
  </xs:sequence>
</xs:complexType>
…
```

2.1.6 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"/>
```

To refresh, the main purpose of *modelReference* extension element is to enable the addition of semantic information to WSDL elements so that this semantic information can be used while matching, and composing Web services. Section 2 provided an overview of how a *modelReference* can be associated with interfaces, operations, simple types, complex types, elements and attributes in a WSDL document for multi-purpose usages such as categorization, representation of preconditions and effects, and at times to simply provide additional semantics to the specific elements.

The next section introduces the second extension attribute in SAWSDL – namely schema mapping to enable adding semantic information to aid in invocation.

2.2 SAWSDL Schema Mapping

Sometimes data elements of a Web service may need to be transformed to invoke another Web service. For example, a Web service that can supply a givenName and a surName may have to concatenate the data values of these two elements to pass to an element called fullName on another Web service. Automatically discovering these types of data transformations among a given set of Web services is a difficult problem. In addition, storing the data transformations required by all possible Web services that a given request can connect with is impractical. A more general approach would be to map the data values of a Web service with semantic data in a semantic model and from the semantic data in a semantic model to the data values of a Web service. This roundtripping transformation information between Web services and semantic models reduces the complexity of having to store transformations between all possible pairs of Web services to a more simpler, canonical form. Therefore, 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 when the structure of the instance data does not correspond to the organization of the semantic data. The mappings are used when mediation code is generated to support invocation of a Web service.

Deleted: To annotate attributes, include

Deleted: h

Formatted: Normal

Formatted: Font: Italic

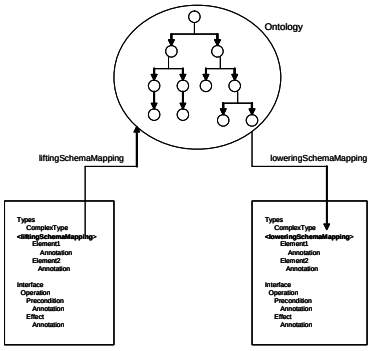


Figure 1 illustrates the idea of lifting and lowering schemaMapping extension attributes pictorially.

While a *modelReference* could, in theory, be used for representing such schema mappings to support data transformations as well, SAWSDL prefers to define a first class object for representing schema mappings rather than overloading a modelReference. Therefore, two additional extension attributes – namely liftingSchemaMapping and loweringSchemaMapping are defined.

This section defines the extension attributes *liftingSchemaMapping* and *loweringSchemaMapping*. They are used to associate a schema type or element with an ontology, as just described. 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 or both together on the same schema element. The following schema excerpt defines the *liftingSchemaMapping* and *loweringSchemaMapping* attributes.

Deleted: 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 when the structure of the instance data does not correspond to the organization of the semantic data. The mappings are used when mediation code is generated to support invocation of a Web service.¶

Deleted: declerations

Deleted: as well

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

The value of the *liftingSchemaMapping* attribute is a list of 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 a 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 list of URI's 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.

This specification provides a mechanism for associating optional schema mapping functions with a global type or global element definition without any restriction on the choice of the schema mapping language.

Just as SAWSDL does not prescribe any particular ontology representation language for specifying modelReferences, similarly, it does not prescribe any particular schema mapping representation language for representing lifting or lowering schema mappings. Users can choose their preferred schema mapping languages such as RDF [RDF], XSLT [XSLT] and XQuery [XQuery]. The following excerpt from the purchase order example shows how XSLT can be used as a schema mapping language to specify associations between XSD elements and concepts in a semantic model. Detailed examples showing mapping using XSLT [XSLT] and XQuery [XQuery] are shown in Appendix A.

```
…
<xs:element name="OrderRequest"

sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/p
urchaseorder#OrderRequest"

sawsdl:liftingSchemaMapping="http://www.w3.org/2002/ws/sawsdl/spec/mapp
ing/Request2Ont.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>
…
```

In order to perform the lifting of the data contained in and XML message, a client processor applies the transformation that can be retrieved from the URI given in the value of sawsdl:liftingSchemaMapping and applies the transformation to those elements where the schema mapping was specified. In the above example, *http://www.w3.org/2002/ws/sawsdl/spec/mapping/Request2Ont.xslt* will be applied to *OrderRequest* elements. If a mapping is specified on a global type definition the mapping definition must only be applied to the elements that are 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.

Deleted: some

Deleted: case

Comment [I4]: <Rama> In the first Face2Face meeting Jacek gave many examples representations of complex types and discussed the implication of associate schemaMappings on each of them. I think we should present those examples here to add more clarity to this discussion. The precedence and overwrite rules are hard to visualize otherwise </Rama>

Deleted: If the element does not specify a mapping, the mapping of its type applies.

The following example illustrates this. In this example, the type 'orderItem' is represented as an element and a schemaMapping is specified on it. The actual complex type of orderItem is defined later in the document and the complex type itself has a schema mapping. In such a case, the schemaMapping that is specified on the element overwrites the one specified on the complex type. This means that only the schema mapping that is given with the URI **http://www.w3.org/2002/ws/sawsdl/spec/mapping/OrderItem2Ont.xslt** applies to the element orderItem. The **http://www.w3.org/2002/ws/sawsdl/spec/mapping/ MyDefOrderItem2Ont.xslt** does not apply. The reason for specifying such an overwrite rule is to avoid any confusion that could arise due to multiple places at which schema mappings can be specified for the same thing in XML schema.

```
<xs:element name="orderItem" type="type"
sawsdl:schemaMapping=http://www.w3.org/2002/ws/sawsdl/spec/mapping/Orde
rItem2Ont.xslt/>

<xs:complexType name="type"
sawsdl:schemaMapping="http://www.w3.org/2002/ws/sawsdl/spec/mapping/MyD
efOrderItem2Ont.xslt>
  <xs:sequence>
      <xs:element ref="part" />
    </xs:sequence>
  <xs:attribute name="att"/>
 </xs:complexType>

<xs:element="part"
sawsdl:schemaMapping=http://www.w3.org/2002/ws/sawsdl/spec/mapping/Mysa
mpleXSLT.xslt type="xs:schema/>
```

When used on an element that represents a WSDL 2.0 Component (top-level *xsd:element*, *xsd:complexType* or *xsd:simpleType*), the loweringSchemaMapping and liftingSchemaMapping extension attributes introduce OPTIONAL properties {lowering schema mapping} and {lifting schema mapping}. The value of either of these properties is a list of URIs taken from the value of the respective attribute. In contrast to the {model reference} property, the absence of the {lifting schema mapping} and {lowering schema mapping} properties *is different from* its presence with an empty value, as mappings on an element must be able to override the mappings specified on the type of the element.

3. 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 services descriptions. The three XML attributes defined in this specification namely *modelReference*, *liftingSchemaMapping* and *loweringSchemaMapping* apply to WSDL 1.1 documents as well with minor changes. For example, in WSDL 1.1, operations are not extensible. To accommodate the annotation of operations, a new element is introduced. The following subsections describes the details.

3.1 SAWSDL attrExtensions Element

Formatted: Font: Bold

Deleted: ¶

Formatted: French (France)

Deleted: All t

Formatted: Font: Italic

Formatted: Font: Italic

Deleted: without modification

Deleted: the

Deleted: descriptions

Formatted: Font: Italic

Deleted: However,

Deleted: in some cases they are applied to different elements in the WSDL document structure and a

Deleted: to facilitate operation annotations

WSDL 1.1 does not allow extension attributes on the *operation* element, Therefore, SAWSDL introduces a new extension element, *attrExtensions* to support semantic annotation of WSDL 1.1 operations. The following XML Schema excerpt defines the *attrExtensions* element.

Deleted: , so

Deleted: this specification

Deleted: d

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

The *attrExtensions* element is for use only with WSDL 1.1 documents. It is not valid in WSDL 2.0 documents, where the *modelReference* attribute should be applied to the *operation* instead. An example for using attrExtensions is provided further below in section 3.2.

Deleted: It provides a general mechanism for adding extension attributes. For SAWSDL it is used to add the sawsdl:modelReference attribute.

3.2 WSDL 1.1 Annotations

Deleted: a

Input and Output

Annotation of XML Schema types in WSDL 1.1 with *modelReference*, *liftingSchemaMapping* or *loweringSchemaMapping* can be accomplished using the same 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 WSDL1.1 which generally corresponds to the WSDL2.0 *interface* structure. One difference is that portTypes do not support inheritance (they have no *extends* relation).

Deleted: or schema

Deleted: that

Faults

In WSDL 1.1, faults are specified as messages that are generated when a particular condition arises. Annotations for fault messages are done just as annotations are done for any other output message using a *modelReference*.

Formatted: Font: Italic

Deleted: .

Operations

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.

```
…
<wsdl:operation name="order">
    <wsdl:input message="OrderRequestMessage"/>
    <wsdl:output message="OrderResponseMessage"/>
```

```
    <sawsdl:attrExtensions
sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/p
urchaseorder#RequestPurchaseOrder"/>
</wsdl:operation>
…
```

portTypes

A *portType* is annotated in the same way as a WSDL 2.0 *interface*.

3.3 Running Example in WSDL 1.1

The same running example used in the other sections of this specification and in Appendix A, is shown here in its WSDL 1.1 form. An important difference is the way messages are defined:

Deleted: differences

- 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* 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.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
  targetNamespace="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order11#"
  xmlns="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order11#"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:sawsdl="http://www.w3.org/2002/ws/sawsdl/spec/sawsdl#"
  xmlns:xsd1="http://example.org/order#">

  <wsdl:types>
    <xs:schema targetNamespace="http://example.org/order#"
elementFormDefault="qualified">
      <xs:element name="OrderRequestT"

sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/p
urchaseorder#OrderRequest"

sawsdl:liftingSchemaMapping="http://www.w3.org/2002/ws/sawsdl/spec/mapp
ing/Request2Ont.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/p
urchaseorder#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>
          <xs:enumeration value="Pending"></xs:enumeration>
          <xs:enumeration value="Rejected"></xs:enumeration>
        </xs:restriction>
      </xs:simpleType>
    </xs:schema>
  </wsdl:types>

  <wsdl:message name="OrderRequestMessage">
        <wsdl:part name="OrderRequest" type="xsd1:OrderRequestT"/>
  </wsdl:message>

   <wsdl:message name="OrderResponseMessage">
          <wsdl:part name="OrderResponse" type="xsd1:Confirmation"

sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/p
urchaseorder#OrderConfirmation"/>
  </wsdl:message>

  <wsdl:portType name="Order"
sawsdl:modelReference="http://example.org/categorization/products/elect
ronics">
    <wsdl:operation name="order">
      <wsdl:input message="OrderRequestMessage"/>
      <wsdl:output message="OrderResponseMessage"/>
      <sawsdl:attrExtensions
sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/p
urchaseorder#RequestPurchaseOrder"/>
      </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

Deleted: EAN

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

4. References

4.1 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.

[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/.

[SOAP]
SOAP Version1.2 Part 1: Messaging Framework, M. Gudgin, M. Hadley, N. Mendelsohn, J-J. Moreau, H. Frystyk Nielsen, Editors. World Wide Web Consortium, 24 June 2003. This version of the "SOAP Version 1.2 Part 1: Messaging Framework" Recommendation is http://www.w3.org/TR/2003/REC-soap12-part1-20030624/. The latest version of "SOAP Version 1.2 Part 1: Messaging Framework" is available at http://www.w3.org/TR/soap12-part1/.

[URI]
UniformResource Identifiers (URI): Generic Syntax, T. Berners-Lee, R. Fielding, L. Masinter, Authors. Internet Engineering Task Force, January 2005. Available at http://www.ietf.org/rfc/rfc3986.txt.

[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.

[XML]
Extensible Markup Language (XML) 1.0 (Third Edition), T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, Editors. World Wide Web Consortium, 4 February 2004. This version of the XML 1.0 Recommendation is http://www.w3.org/TR/2004/REC-xml-20040204/. The latest version of "Extensible Markup Language (XML) 1.0" is available at http://www.w3.org/TR/REC-xml.

[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.

[XMLSchema Part 2: Datatypes]
XML Schema Part 2: Datatypes, P. Byron and A. Malhotra, Editors. World Wide Web Consortium, 28 October 2004. This version of the XML Schema Part 2 Recommendation is http://www.w3.org/TR/2004/REC-xmlschema-2-20041028. The latest version of XML Schema Part 2 is available at http://www.w3.org/TR/xmlschema-2.
[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/.

4.2 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.
[ODM]
Ontology Definition Metamodel. Revised Submission to OMG/RFP ad/2003-03-40, Jan. 2005.Available at http://www.omg.org/docs/ad/05-01-01.pdf.
[UML]
Unified Modeling language (UML) Version 2.0. Object Management Group. Available at http://www.omg.org/technology/documents/formal/uml.htm.
[UNSPSC]
The United Nations Standard Products and Services Code. United Nations Development Programme. Available at http://www.unspsc.org/.
[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.

A. An Example (Non-Normative)

This is an example WSDL document that is annotated with semantic information to give the reader an example of what was explained in the specification so far. The semantic annotations that have been explained in section 2 drew from this example.

The ontology used in this example is:

Deleted: The purpose of the ontologies is purely to illustrate the concepts outlined within this specification. The relevance of the concepts referenced might be increased in subsequent versions of the specification.¶

Deleted: is explained in the rest

Deleted: of the

Deleted: document

- A purchase order ontology: http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#

The ontology given here is purely for illustrative purpose only and cannot be treated as an authoritative ontology for purchase order process.

Formatted: Justified

The complete wsdl file is given below and is available as a separate file at http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#.

```
<?xml version="1.0" encoding="UTF-8"?>
<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/2006/01/wsdl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:sawsdl="http://www.w3.org/2002/ws/sawsdl/spec/sawsdl#">

  <wsdl:types>
    <xs:schema targetNamespace="http://example.org/order#"
elementFormDefault="qualified">
      <xs:element name="OrderRequest"

sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/p
urchaseorder#OrderRequest"

sawsdl:liftingSchemaMapping="http://www.w3.org/2002/ws/sawsdl/spec/mapp
ing/Request2Ont.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/p
urchaseorder#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"
sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/p
urchaseorder#OrderConfirmation"/>
      <xs:simpleType name="Confirmation">
        <xs:restriction base="xs:string">
          <xs:enumeration value="Confirmed"></xs:enumeration>
          <xs:enumeration value="Pending"></xs:enumeration>
          <xs:enumeration value="Rejected"></xs:enumeration>
        </xs:restriction>
      </xs:simpleType>
    </xs:schema>
```

Deleted: EAN

```
  </wsdl:types>

  <wsdl:interface name="Order"
sawsdl:modelReference="http://example.org/categorization/products/elect
ronics">
    <wsdl:operation name="order"
pattern="http://www.w3.org/2006/01/wsdl/in-out"

sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/p
urchaseorder#RequestPurchaseOrder">
      <wsdl:input element="OrderRequest"/>
      <wsdl:output element="OrderResponse"/>
    </wsdl:operation>
  </wsdl:interface>
</wsdl:description>
```

In this example, we present a simple Web service for an item purchase order. The semantic concepts and their relationships are modeled in an OWL ontology. The inputs, outputs of the ProcessPurchaseOrder service are annotated within the respective schema definition.

A.1 Specifying Schema Mapping Using XSLT

The XSLT transformation is work-in-progress and known to be incomplete.

Deleted:

Deleted:

```
<?xml version='1.0' ?>
<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://example.org/ontologies/purchaseorder#">
  <xsl:output method="xml" version="1.0" encoding="iso-8859-1"
indent="yes" />

  <!-- XSLT will receive the node po:OrderRequest as root -->
  <xsl:template match="/">
    <rdf:RDF>
      <POOntology:OrderRequest rdf:ID="{generate-id(.)}">
        <xsl:for-each select="//po:orderItem">
          <POOntology:has_items rdf:resource="#{generate-id(.)}"/>
        </xsl:for-each>
      </POOntology:OrderRequest>
      <xsl:apply-templates select="/*/po:orderItem" />
    </rdf:RDF>
  </xsl:template>

  <xsl:template match="po:orderItem">
    <POOntology:Item rdf:ID="{generate-id(.)}">
      <POOntology:has_UPCCode><xsl:value-of select="po:UPC"
/></POOntology:has_UPCCode>
      <POOntology:has_Quantity><xsl:value-of select="@quantity"
/></POOntology:has_Quantity>
    </POOntology:Item>
  </xsl:template>
```

Deleted: EAN

Deleted: EAN

Deleted: EAN

```
</xsl:transform>
```

Below the result of the xslt transformation applied to a sample message.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<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://example.org/ontologies/purchaseorder#">
   <POOntology:OrderRequest rdf:ID="d1">
      <POOntology:has_items rdf:resource="#d1e6"/>
      <POOntology:has_items rdf:resource="#d1e12"/>
   </POOntology:OrderRequest>
   <POOntology:Item rdf:ID="d1e6">
      <POOntology:has_UPCCode>912898437732</POOntology:has_UPCCode>
      <POOntology:has_Quantity>2</POOntology:has_Quantity>
   </POOntology:Item>
   <POOntology:Item rdf:ID="d1e12">
      <POOntology:has_UPCCode>124423231231</POOntology:has_UPCCode>
      <POOntology:has_Quantity>4</POOntology:has_Quantity>
   </POOntology:Item>
</rdf:RDF>
```

Deleted: EAN

Deleted: EAN

Deleted: EAN

Deleted: EAN

Formatted: English (U.S.)

A.2.2 Specifying Schema Mapping Using XQuery

We will provide this in future versions of the specification

B. Mapping Choices

This specification does not prescribe any specific schema mapping language for use in describing schemaMappings. However, as domain models like UML, OWL and XSD are represented in XML, the examples in the document represent mappings either as transformations using XSLT or queries using XQuery. Using these mappings from XML schema elements to concepts in a domain model, it is possible to map instances of Web service schemas from one representation to another. The process of capturing mappings can be complicated by heterogeneities at various levels and XQuery or XSLT might not be suited for all cases. The annotation mechanism defined in this specification can refer to mappings expressed in other mapping languages as well since it is agnostic to the mapping language used.

Deleted: specify a

Deleted: single

Deleted: to use

Deleted: xml

Table 1 illustrates the schema/data conflicts that arise during mapping schemas. This is provided as a background reading for the reader of the specification.

Deleted: most

Deleted: representation languages are capable of handling and how instances of one schema can be mapped to instances of another using a common ontology

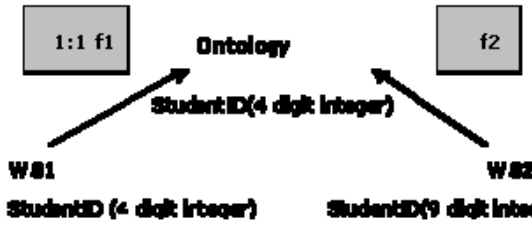
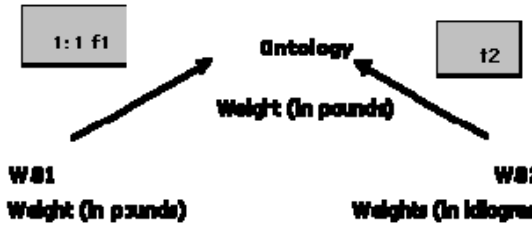
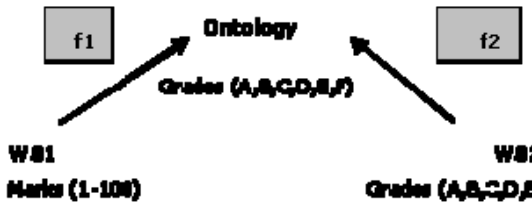
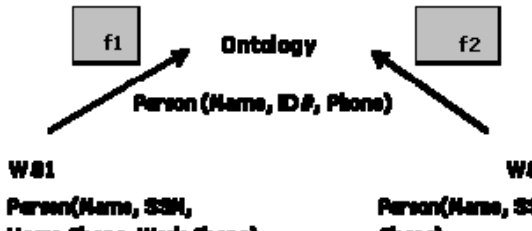
Schema/Data Conflicts	Description/ Example	Nature of mapping function
Data Representation conflict	Different data types / representations 	The mapping function ft will largely depend on application / domain requirements. *Note: While mapping in the direction of f2 can be well defined, $f2^{-1}$ can not.
Data Scaling conflict	Representations using different units and measures 	The mapping function ft or its inverse $f2^{-1}$ can be automatically generated using a look up table and are well defined
Data Precision conflict	Represented using different precisions 	The mapping function ft will largely depend on application requirements. Example: A (81-100); B (61-80); C (41-60); D (21-40); E (:-20)
Schema Isomorphism conflict	Schema of similar elements have different number of attributes 	The mapping function ft will largely depend on application requirements. ft may be defined as 1. Home Phone (WS1) -> Phone(Ontology) or 2. Work Phone (WS2) -> Phone(Ontology) Note: While mapping in the direction of ft can be well defined, ft^{-1} can not.

Table 1: Possible schematic / data conflicts between XML input/output messages

Deleted: xml

* Although the mapping function is well defined in one direction, from the WSDL element to the Ontology concept, it is not well defined in the reverse direction. Although converting a 5 digit StudentID to a 9 digit StudentID is conceivable through use of a look up table, the transformation is not a well defined function in itself.

Legend: WS1, WS2 denote Web services 1 and 2 and f1 and f2 denote mapping functions from the WSDL elements to the ontology.

C. XML Schema for Semantic Annotations for WSDL

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

```
<?xml version="1.0" encoding="UTF-8"?> <xs:schema
  targetNamespace="http://www.w3.org/2002/ws/sawsdl/spec/sawsdl#"
  xmlns="http://www.w3.org/2002/ws/sawsdl/spec/sawsdl#"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:wsdl="http://www.w3.org/2006/01/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:schema>
```

D. An Example Categorization Model (Non-Normative)

One purpose for annotating services at the interface level is to help enable dynamic discovery. This is possible when services are published, cataloged and annotated with semantics. The *modelReference* mechanism on an interface can be used to add categorization information to a WSDL document which could be used while publishing services in registries such as Universal Description, Discovery and Integration (UDDI). Users can choose any categorization of their choice such as NAICS [NAICS], UNSPSC [UNSPSC] and GICS [GICS]. This categorization of services aids in service discovery by narrowing the range of candidate services. Additionally, it can constitute the effective categorization when the service is made available via Web Services Inspection Language [WSIL] or some other solution-specific means. Service categorization is also aimed at supporting specialized taxonomies of middleware or utility services such as mediators. The category *modelReference* can be used to ensure that there is basic and high-level categorization information about a service.

In the following example, a *modelReference* points to a concept in an ontology that gives three pieces of information, the name of the category, the taxonomy value (category code) of the category and the URI of the taxonomy. An example of instance data that would be referenced by an interface annotation is shown here.

Deleted: is

```
<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="urn: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#Cat
egory"/>
  <rdf:Property
rdf:about="http://www.w3.org/2002/ws/sawsdl/spec/ontology/interface#has
Value">
    <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#use
sTaxonomy">
    <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>
```

E. 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 by 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.), Nicholas Gibbins (University of Southampton), James Hendler(Maryland Information and Network Dynamics Lab at the University of Maryland), 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), Jianhui Li (Chinese Academy of Sciences), Terry Payne (University of Southampton), Carlos Pedrinaci (The Open University), Eric Prud'hommeaux (W3C), Brahmananda Sapkota (DERI Galway at the National University of Ireland, Galway, Ireland), Amit Sheth (Semagix, Inc.), Claudio

Venezia (Telecom Italia SpA), Tomas Vitvar (DERI Galway at the National University of Ireland, Galway, Ireland).

The working group also recieved significant help from John Miller (University of Georgia).

F. Change Log (Non-Normative)

Table F-1. Summary of Semantic Annotations for WSDL Specification Changes

Date	Author	Description
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).

Table F-1. Summary of Semantic Annotations for WSDL Specification Changes

Date	Author	Description
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.

G. Editors' Todo List (Non-Normative)

Table G-1. Summary of Editors' Semantic Annotations for WSDL Specification Work Items

Type	Description
Editorial	Need to update examples so a single example is built up from the introduction though the appendix.
Editorial	Example of an XSLT schema mapping function needs to be updated and an XQuery example needs to be added.