Design and Management of Semantic Web Services using Conceptual Model

Martin Necasky, Jaroslav Pokorny
Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic
{martin.necasky, jaroslav.pokorny}@mff.cuni.cz

Abstract

There are emerging technologies such as SAWSDL or WSMO that extend the current Web Services technologies to so called Semantic Web Services by combining the structural and semantic descriptions of Web services. In this paper, we identify problems that can arise when using these technologies for the design and management of structural and semantic descriptions of Web services. We show that using a conceptual model instead of these technologies can help to solve these problems. We also show how to automatically derive the structural and semantic description represented with SAWSDL from the conceptual level because this representation brings other advantages. The derivation of WSMO representation is also possible.

1. Introduction

Web Services (WS) is a set of technologies for an implementation of the service oriented architecture (SOA) on the Web. Services, implemented as WS, communicate by exchanging XML messages. Each service provides an interface that describes its input and output messages. Currently, WS Description Language (WSDL) [8] is used as the interface description language encapsulating XML Schema [6] for describing the syntactical structure of messages. However, it can not describe the semantics of the messages. The semantics is handled implicitly by the provider and clients. Therefore, the processes of automatic discovery and composition of services are hard to solve. Recently, researchers have proposed an extension of the current WS called *Semantic Web Services* (SWS) that should help to automate these processes by describing the semantics with ontologies. The provider publishes XML schemes, called *source schemes*, which are the *structural description* of the messages, and binds the structural description with an ontology, called *target ontology*, that provides the *semantic description*. The binding is called *grounding* and specifies how to translate the messages between the structural and semantic data representation.

Grounding technologies are summarized in [3]. The most recent are SAWSDL [9] and WSMO [1]. SAWSDL extends WSDL and allows to specify the grounding in the structural description, i.e. in XML schemes. On the other hand, WSMO allows to specify the grounding in the semantic description, i.e. in ontologies. The third possibility, the grounding externalized from both descriptions, has not been addressed yet as discussed in [3].

We suppose that providers and clients of SWS will form communities and will share ontologies inside but not fully between the communities. Figure 1 shows a motivating example. There is a *Sales* service with two operations *Process Order* and *Process Payment*. The structure of input and output messages is given by the XML schemes shown at the figure. Because the company wants to provide the customers with the communication independent of this structure SWS are used. The company provides the structural description and the grounding to several target ontologies given

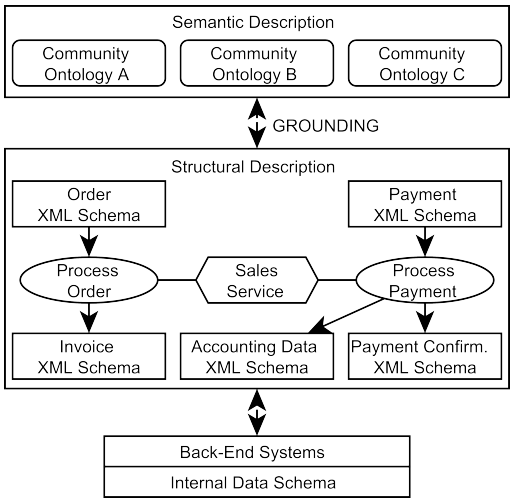

Figure 1. Sales Service Architecture

1-4244-0674-9/06/$20.00 ©2006 IEEE.

by different customer communities. Suppose that each customer provides the grounding for the ontology of its community. The figure shows ontologies of three hypothetical communities. Suppose that a customer from *Community A* sends an order message to *Sales Service*. First, the message having the structure given by the customer is translated to the semantic repreasentation given by *Community Ontology A* using the grounding provided by the customer and then translated to the structural representation given by *Order XML Schema* of *Sales Service* using the grounding provided by the company. In this structural representation, the order message can be delivered to and processed by *Sales Service*. The translations are performed automatically.

Contribution. SAWSDL and WSMO require to provide the grounding for each source schema and target ontology separately. Therefore, there can be up to *M:N* groundings whose design and management is decentralized to more different structural (in the case of SAWSDL) or semantic (in the case of WSMO) descriptions. If there is the same concept repeated in more XML schemes the provider must specify the grounding for this concept repeatedly. This is hard to design and manage. On the other hand, the provider usually has an internal data schema describing how the data is represented by the back-end systems. This schema is usually conceptual. The structural description of the service interfaces usually results from this internal schema. In this paper, we show how to utilize this conceptual schema for grounding externalized from both structural and semantic descriptions which allows to design and manage the descriptions and grounding of all provider's services sharing the same data domain in one conceptual schema. This possibility has not been addressed yet by the recent technologies. On the other hand, it must be possible to derive SAWSDL or WSMO grounding from the externalized grounding because these technologies have also advantages, as shown in [3]. For our approach, we employ a conceptual model for XML called *XSEM* [4].

The rest of the paper is organized as follows. In Section 2 we describe the XSEM model. In Section 3 we show how to use XSEM as the externalized grounding and how to derive its SAWSDL representation. We conclude in Section 4.

2. XSEM Model

XSEM is a conceptual model for XML data. It is composed of two parts called *XSEM-ER* and *XSEM-H*.

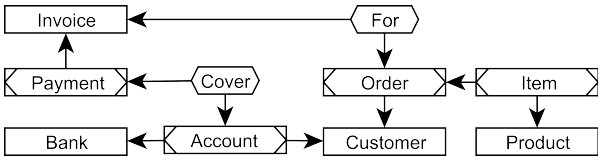

Figure 2. XSEM-ER Schema

XSEM-ER is used for modelling the semantics of the data. It is based on E-R. Because of special features of XML such as irregular structure, ordering, and mixed content, it provides some extending modelling constructs. The hierarchical structure of the data is not important here, only the semantics is modelled. For a more detail we refer to [4]. Figure 2 shows an XSEM-ER schema of the internal data representation of the company providing *Sales Service*. There are strong entity types, such as *Product*, modelling stand-alone real world objects. There are also weak entity types, such as *Item*, modelling real world objects whose existence depends on other objects that are modelled by entity types connected to the weak entity type as so called *determinants*. For example, *Item* has two determinants, *Order* and *Product*. Moreover, there are relationship types, such as *For,* modelling relationships between two or more objects that are modelled by entity types connected to the relationship type as so called *participants*. For example, *For* has two participants, *Invoice* and *Order*. Attributes of entity and relationship types are not shown at the figure.

With XSEM-H we specify how the components from the XSEM-ER are organized in the hierarchical XML documents. XSEM-H schemes are called *hierarchical views* on the XSEM-ER schema. From each hierarchical view an XML schema is derived. The hierarchical view serves as a binding between the conceptual XSEM-ER schema and the derived XML schema. Figure 3 shows three hierarchical views on the XSEM-ER schema from Figure 2. From the views the XML schemes *Order, Invoice,* and *Payment* providing the structural description of *Sales Service* are derived.

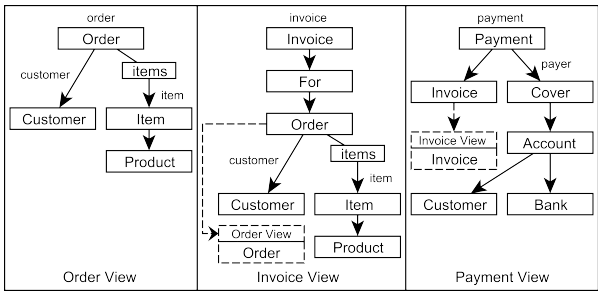

Figure 3. XSEM-H Hierarchical Views

3. Architecture

Figure 4 shows an overall architecture for the design and management of *Sales Service* utilizing XSEM. A human designer first designs an XSEM-ER schema as an internal conceptual schema (it can be an existing E-R schema as well) (1). Then he or she designs the hierarchical views on the XSEM-ER schema as the conceptual description of the *Sales Service* interface (2). The structural description, i.e. the XML schemes, is then derived automatically from the hierarchical views (3). The grounding is not specified directly between the XML schemes and the target ontologies. Instead it is specified only once for each target ontology on the conceptual level, i.e. between the XSEM-ER conceptual schema and the target ontology (4). The grounding represention with SAWSDL or WSMO can be derived automatically from this central conceptual grounding (5) (in the rest of the paper we comprehend only SAWSDL because of the lack of the space). The solid lines at the figure denote a manual work of the designer and the dotted lines denote an automatic derivation. If a change must be made in the internal data schema, it is made in the XSEM-ER conceptual schema and then propagated automatically to the affected XML schemes and groundings. Also changes in the target ontologies can be propagated automatically to the grounding. In the following two subsections we describe the conceptual level grounding and the derivation of its SAWSDL representation in detail.

3.1. Conceptual Level Grounding

The grounding on the conceptual level is not provided directly for the XSEM-ER schema but for an ontology that is automatically derived from the XSEM-ER schema as shown at Figure 4. We call this ontology *derived ontology*. Suppose OWL as an ontology language. We represent each entity and relationship type from the XSEM-ER schema as an OWL class. If an entity type T_1 is a determinant of a weak entity type or a participant of a relationship type T_2 we add an OWL object property with the domain C_1 and range C_2 where C_1 and C_2 are the OWL classes representing T_1 and T_2, respectively. Simple attributes of entity and relationship types are represented by OWL data type properties. A complex attribute, i.e. composed of other attributes, is represented by a class encapsulating the attributes composing the complex attribute and connected with the corresponding class by an object property. Even though we can not

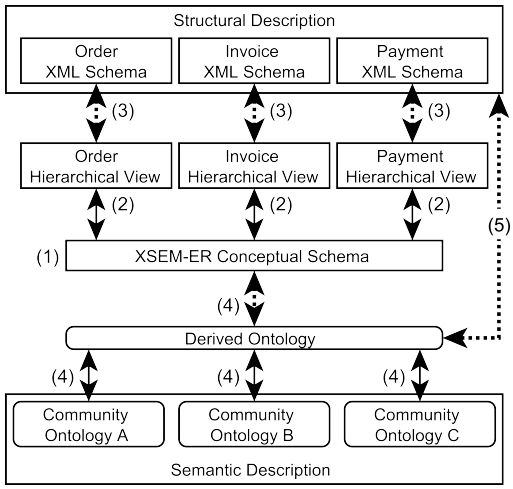

Figure 4. Semantic Web Services Design Architecture

reconstruct the XSEM-ER schema back from the derived ontology (for example a class can represent an entity type, relationship type, or complex attribute) it contains sufficient information for the grounding.

Example 1 shows a part of the derived ontology for the XSEM-ER schema from Figure 2 in the triple notation. The names of classes are given by the corresponding types in the XSEM-ER schema. For

```
:Customer rdf:type owl:Class .
:has_name rdf:type owl:DataTypeProperty ;
        rdfs:domain :Customer;
        rdfs:range xs:string .
:Product rdf:type owl:Class .
:has_code rdf:type owl:DataTypeProperty ;
        rdfs:domain :Product;
        rdfs:range xs:string .
:has_title rdf:type owl:DataTypeProperty ;
        rdfs:domain :Product;
        rdfs:range xs:string .
:Order rdf:type owl:Class .
:has_ship_date rdf:type owl:DataTypeProperty ;
        rdfs:domain :Order;
        rdfs:range xs:date .
:has_customer rdf:type owl:ObjectProperty ;
        rdfs:domain :Order;
        rdfs:range :Customer .
:has_item rdf:type owl:ObjectProperty ;
        rdfs:domain :Order;
        rdfs:range :Item .
:Item rdf:type owl:Class .
:has_product rdf:type owl:ObjectProperty ;
        rdfs:domain :Item;
        rdfs:range :Product .
:has_order rdf:type owl:ObjectProperty ;
        owl:inverseOf :has_item .
:has_price rdf:type owl:DataTypeProperty ;
        rdfs:domain :Item;
        rdfs:range xs:decimal .
```

Example 1. Derived Ontology

example there is a class *Customer* resulting from the entity type *Customer*. The name of a property representing a participant or determinant is the name of the participant or determinant, respectively, preceded by 'has_'. For example, because the entity type *Customer* is a determinant of *Order* we derive an object property *has_customer*. Properties representing attributes are named in the same way.

With the XSEM-ER schema translated to the derived ontology a grounding between the XSEM-ER schema and a target ontology is a mapping between the derived and target ontology. A mapping for each target ontology must be specified. For this we can profitably utilize existing languages for a mapping between ontologies such as [5]. In [2] a survey of mapping languages is provided. We can also use OWL constructs for a basic mapping. Therefore, we do not need to provide any extending constructs to ontology languages. Provided the mapping between the derived and target ontology we can find for each component in the XSEM-ER schema a corresponding concept in the target ontology, i.e. the mapping serves as the conceptual level grounding.

3.2. SAWSDL Grounding Representation

SAWSDL extends XML Schema and WSDL with three attributes. The attribute *modelReference* is used to specify an association between an XML schema component and a concept in a semantic model. The attributes *liftingSchemaMapping* and *lowering-SchemaMapping* are used to specify XSLT mappings between the structural and semantic representations.

Firstly, we show the derivation of XML schemes from hierarchical views. If a node is the root node or the edge going to the node has a label then it is translated to a complex type definition. Otherwise it is translated to a group. The content of the complex type or group, respectively, is given by the attributes of the node and the edges going from the node. Each attribute and labeled edge is translated to an element declaration with the corresponding type. An edge without a label is translated to a reference to the group representing the child node of the edge. Example 2 shows an XML schema derived from the first hierarchical view at Figure 3. There is an element declaration *order* for the root node from the view with the corresponding complex type definition *Order*. It contains an element declaration corresponding to an attribute *shipDate* of *Order* and element declarations corresponding to the edges going from the root node. These element declarations are named with the labels of the edges, i.e. *customer* and *item*. The translation continues recursively to the descendant nodes. The only difference is in the translation of the edge going from *Item* to *Product* which has no label. This can not be translated to an element declaration because we do not have a name for it. Therefore, we merge the content corresponding to *Product* with the content corresponding to *Item*. However, we need to distinguish which element declarations belong to *Product* and which to *Item* on the schema level. Therefore, we use a mechanism of XML Schema groups as shown by the example.

Secondly, we show the derivation of the grounding for XML schemes derived from hierarchical views. There are two possibilities. The simpler possibility is to derive the grounding to the derived ontology. Then we can use a semantic reasoner to dynamically translate between the semantic representations given by the derived and target ontology using the mapping between them. The second possibility is to derive the

```
<xs:element name="order" type="Order"/>
<xs:complexType name="Order"
        sawsdl:modelReference="d_ont#Order"
        sawsdl:liftingSchemaMapping="Order.xslt">
 <xs:sequence>
  <xs:element name="shipDate" type="xs:date"
        sawsdl:modelReference="d_ont#has_ship_date"/>
  <xs:element name="customer" type="Customer"
        sawsdl:modelReference="d_ont#has_customer"/>
  <xs:element name="item" type="Item"
        maxOccurs="unbounded"
        sawsdl:modelReference="d_ont#has_item"/>
 </xs:sequence>
</xs:complexType>
<xs:complexType name="Customer"
        sawsdl:modelReference="d_ont#Customer">
  <xs:sequence>
    <xs:element name="name" type="xs:string"
        sawsdl:modelReference="d_ont#has_name"/>
    <xs:group name="Product" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="Item"
        sawsdl:modelReference="d_ont#Item">
  <xs:sequence>
    <xs:element name="price" type="xs:decimal"
        sawsdl:modelReference="d_ont#has_price"/>
    <xs:group name="Product" />
  </xs:sequence>
</xs:complexType>
<xs:group name="Product"
        sawsdl:modelReference="d_ont#Product">
 <xs:sequence>
   <xs:element name="code" type="xs:string"
        sawsdl:modelReference="d_ont:has_code" />
   <xs:element name="title" type="xs:string"
        sawsdl:modelReference="d_ont:has_title" />
 </xs:sequence>
</xs:group>
```

Example 2. XML Schema with SAWSDL Grounding

grounding for each target ontology separately. In this paper we show only the first possibility.

The SAWSDL grounding of an XML schema to the derived ontology consists of two parts. Firstly, each component in the XML schema must be bounded with the corresponding concept from the derived ontology using *modelReference*. Each complex type declaration or group resulting from a node in the hierarchical view is bounded with the class in the derived ontology corresponding to the node. For example, the complex type declaration resulting from the node *Order* is bounded with the class *Order* from the derived ontology at Example 2. Further, each element definition resulting from an attribute or edge is bounded with the corresponding property in the ontology. For example, the element definition *shipDate* is bounded with the property *has_ship_date* and *customer* is bounded with *has_customer*.

Secondly, the lifting and lowering XSLT mappings are generated automatically for the XML schema from the corresponding hierarchical view. We show how to generate the lifting schema. It transforms source XML messages from the structural to semantic representation given by the derived ontology. The semantic representation is serialized in RDF/XML. For each node in the hierarchical view we create an XSLT template matching the corresponding elements in the source XML messages. This template transforms the matched elements to their semantic representation. We start with a template matching the root of the source XML message. Its semantic representation in the RDF/XML serialization is an element with the name of the corresponding class from the derived ontology. Each property is represented by a child element with the name of the property. If it is a data type property, its value is extracted with an XPath expression from the source XML message. If it is an object property its value is constructed recursively.

Example 3 shows a sample template from the XSLT mapping derived from the first hierarchical view at Figure 3. It matches the root element *order*. According to the derived ontology, the element is transformed to its semantic representation serialized in RDF/XML as an element *Order*. The child elements of *order,* i.e. *shipDate*, *customer*, and *item*, correspond to properties that are serialized to the child elements of the element *Order,* i.e. *has_ship_date*, *has_customer*, and *has_item*, respectively. The value of the first property is retrieved with an XPath expression. The value of the other two properties are reconstructed with corresponding templates that are derived from the hierarchical view in the same way.

```
<xsl:template match="/order">
  <Order>
    <has_ship_date><xsl:value-of select="shipDate"/>
    </has_ship_date>
    <has_customer>
      <xsl:apply-templates select="customer"/>
    </has_customer>
    <has_item><xsl:apply-templates select="item"/>
      </has_item>
  </Order>
</xsl:template>
```

Example 3. Structural to Semantic Mapping

4. Conclusions

In this paper we showed how to use a conceptual model for XML data as a technology for binding structural and semantic descriptions of Semantic Web Services. We showed that the specification of the binding on the conceptual level has advantages when designing and managing Semantic Web Services. We also showed how to automatically translate the conceptual level binding to the representation using existing technologies, concretely SAWSDL, because these technologies have other advantages that may be required by clients and providers of services.

Acknowledgement. This paper was supported by the National programme of research (Information society project 1ET100300419).

[1] C. Feier, J. Domingue, ”WSMO Primer”, Final Draft, April 2005*, http://www.wsmo.org/TR/d3/d3.1/v0.1/.*

[2] Y. Kalfoglou and W.M. Schorlemmer, ”Ontology Mapping: The State of the Art”, *Semantic Interoperability and Integration*, Schloss Dagstuhl, Germany, 2005.

[3] J. Kopecky, D. Roman, M. Moran, D. Fensel, "Semantic Web Services Grounding", *AICT/ICIW*, IEEE Computer Society, 2006, p. 127.

[4] M. Necasky, "XSEM - A Conceptual Model for XML", *In Proc. of Asia Pacific Conference on Conceptual Modelling*, Ballarat, Australia. CRPIT, 67, 2007, pp. 37-48.

[5] F. Scharffe, J. Bruijn, "A language to specify mappings between ontologies", *Proceedings of the First International IEEE Conference on Signal-Image Technology and Internet-Based Systems*, 2005, pp. 267-271.

[6] W3C. "XML Schema Part 0: Primer Second Edition", Recommendation, October 2004.

[7] W3C. "XSL transformations (XSLT) version 2.0", Candidate Recommendation, November 2005.

[8] W3C, ”Web Services Description Language (WSDL) Version 2.0”, Working Draft, March 2007.

[9] W3C, ”Semantic Annotations for WSDL and XML Schema”, Candidate Recommendation, January 2007.