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 Semantic Web Services and 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 description of Semantic Web Services represented with existing technologies, concretely with SAWSDL, from the conceptual level because this representation brings other advantages.

Categories and Subject Descriptors
H.3.5 [On-line Information Services]: Web-based services; D.2.12 [Interoperability]: Data mapping

General Terms
Semantic Web Services

Keywords
grounding, conceptual modelling, data transformation

1. INTRODUCTION
Web Services (WS) is a set of technologies that allow to implement 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 an interface description language encapsulating XML Schema [6] for describing the syntactical structure of messages. However, it does not provide any constructs for describing the semantics of the messages. The semantics is handled implicitly by service providers and clients. Therefore, the problems of automatic discovery and composition of web 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. A provider of a web service publishes XML schemes, called source schemes, which are the structural description of messages that can be exchanged with the web service, 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.

The most recent grounding technologies are SAWSDL [9] and WSMO [1]. SAWSDL extends WSDL and allows to specify the grounding in the structural description, i.e. in XML schemes. The idea is to annotate parts of a WSDL description with concepts from the target ontology. On the other hand, WSMO allows to specify the grounding in the semantic description, i.e. in ontologies. The idea is to specify how is a given concept in the target ontology represented by a source XML schema. Grounding technologies are summarised in a more detail in [3]. The third possibility is a grounding externalized from both descriptions, i.e. from the structural and semantic description. However, this possibility has not been addressed yet as discussed in [9].

To introduce the problem, we try to solve in this paper, we start with an example. We suppose a company that provides its customers with two web services ProcessOrder and ProcessPayment. The web services are displayed at Figure 1. ProcessOrder service has an operation processOrder that receives an order request as an input, processes the order, and sends an order confirmation as an output. The structure of the messages is described by XML schemes shown at the figure, i.e. OrderRequest and OrderConfirm XML schema, respectively. ProcessPayment service has an operation processPayment. It receives a payment for a processed order as an input, processes the payment, and sends a payment confirmation as an output. Again, the structure of the messages is described by XML schemes shown at the figure.

In the example, SWS are used to provide customers with communication independent of the structure described by the XML schemes. We suppose that customers form three independent communities and customers in one community share the same ontology to describe the semantics of their data. However, ontologies of two different communities are different and are not mapped on each other. The company provides the structural description of its web services (i.e. source XML schemes). It must also provide groundings of its source XML schemes to the target community ontologies.
These groundings are displayed with dashed lines at Figure 1. Each customer provides the grounding of its own structural description to the ontology of its community. Suppose that a customer from Community A sends an order request message to ProcessOrder service. The message has the structure required by the customer and is translated to the semantic representation given by Community Ontology A. For this translation the grounding provided by the customer is used. Then it is translated to the structural representation given by OrderRequest XML schema of ProcessOrder service. For this the corresponding grounding provided by the service provider is used. In this structural representation, the order request message can be delivered to ProcessOrder service. Conversely, the output order confirmation message is translated from the service provider’s representation to the customer’s representation in the same way. 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 \( M \times N \) groundings (\( M \) and \( N \) are the number of source schemas and target ontologies, respectively) whose design and management is decentralised to more structural (in the case of SAWSDL) or semantic (in the case of WSMO) descriptions. In our example at Figure 1 there is \( 4 \times 3 \) groundings. 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. In this paper, we show how to utilise a conceptual model to overcome these problems. We show that a common conceptual schema of provider’s services can be used for the grounding externalized from both structural and semantic description. It enables to design and manage the structural descriptions and groundings at one place in one conceptual schema and reduces the number of required groundings to \( N \). Such a possibility has not been addressed yet by recent research. On the other hand, it must be possible to derive SAWSDL or WSMO representation of groundings because these technologies have other advantages important at run-time as shown in [3]. For our approach, we employ a conceptual model for XML called XSEM.

The rest of the paper is organised as follows. In Section 2 we describe the XSEM model. In Section 3 we show how to use XSEM for 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: XSEM-ER and XSEM-H. XSEM-ER is used for modelling the semantics of the data. It extends the E-R model with new constructs for modelling special features of XML data. The hierarchical structure of the data is not important here. Because of the lack of the space we refer to [4] where the extending modelling constructs are described in detail. In this paper, we describe only the basic constructs. Figure 2 shows an XSEM-ER schema which is the common conceptual schema of the XML schemes from our example. 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. This is modelled by entity types connected to the weak entity type as so called determinants. For example, Item has two determinants, Order and Product. It means that an existence of an item depends on the ordered product and the order. Moreover, there are relationship types, such as Payment, modelling relationships between two or more objects that are modelled by entity types connected to the relationship type as so called participants. For example, Payment has two participants, CreditCard and Order. Both entity and relationship types have attributes. For example, Product has attributes number and title.

An XSEM-ER schema describes the semantics of the data. It does not describe hierarchical structure. This is left to the designer who derives one or more hierarchical schemes from the XSEM-ER schema. For this the designer uses XSEM-H. An XSEM-H schema is a hierarchical view on the XSEM-ER schema. It does not add any semantics. There can be several XSEM-H hierarchical views on the same XSEM-ER schema. It allows to specify more different hierarchical structures of the same data. From each XSEM-H hierarchical view an XML schema is derived automatically as we show later in this paper. Therefore, an XSEM-H view is a binding between the XSEM-ER schema and the derived XML schema. In our example we need to model the four XML schemes OrderRequest, OrderConfirm, Payment, and PaymentConfirm. The semantics is modelled by the XSEM-ER schema at Figure 2. The hierarchical structure is modelled by the XSEM-H views OrderRequestView, OrderConfirmView, PaymentView, and PaymentConfirmView, respectively, at Fig. 3.

An XSEM-H view is a tree with oriented edges. Each node in the tree represents an entity or relationship type from the XSEM-ER schema, lists zero or more attributes of the represented type, and can have assigned a label. The label determines names of the corresponding elements in XML data. For example, the root of OrderRequestView represents the weak entity type Order. It lists the attributes shipDate and shipAddr and has assigned a label orderRequest. It does
not list the attributes code and price because they are not required for the order request and are assigned after the processing of the request. The root node specifies that each order XML request must have a root element orderRequest that represents an order. Further, the element orderRequest must contain child elements shipDate and shipAddr representing the corresponding attributes of the order. Edges specify hierarchical structure. Each edge connects two nodes and is oriented from a parent to child node. In a basic form, it can connect a node representing a relationship or weak entity type with a node representing its participant or determinant, respectively. For example, there are two edges OrderRequestView going from the root node. The first edge goes to a node representing Customer which is a participant of Order. The second edge goes to a node representing Item that has Order as a participant. The edges specify that the root element in an order XML request contains a child element customer representing the customer who requests the order and a list of child elements item representing items of the order. These child elements are preceded by the child elements shipDate and shipAddr.

There can also be n-ary relationship and weak entity types in an XSEM-ER schema and there are much more possibilities of how to represent them in hierarchical structure. However, we do not describe these possibilities in this paper. For a more detail we refer to [4]. There is also a formalism for the specification of hierarchical structure called hierarchical projections. It is described in detail in [4].

3. ARCHITECTURE

Figure 4 shows an overall proposed architecture for the design and management of the services ProcessOrder and ProcessPayment utilizing XSEM. A human designer first designs a common conceptual XSEM-ER schema (it can be an existing E-R schema as well) (1). Then he or she derives manually the hierarchical views on the XSEM-ER schema as the conceptual descriptions of the XML schemes that provide the structural description of the services (2). These XML schemes are derived automatically from the hierarchical views (3). The grounding is specified for each target ontology on the conceptual level only once. We call this grounding conceptual grounding. Conceptual grounding is not specified directly between the XSEM-ER conceptual schema and the target ontology. Firstly, an ontology called derived ontology is derived automatically from the XSEM-ER schema (4). The conceptual grounding for each target ontology is then specified as a mapping between the derived and target ontology (5). The SAWSDL or WSMO representation of groundings of XML schemes to target ontologies shown at Figure 1 (the dashed lines) can then be derived automatically from the conceptual grounding (in the rest of the paper we comprehend only SAWSDL because of the lack of the space). The solid lines at Figure 4 denote manual work of the designer and the dotted lines denote an automatic derivation. The architecture reduces the required number of groundings from 4 × 3 to 3 conceptual groundings.

The process of designing XSEM-ER schemes and derivation of XSEM-H views from the XSEM-ER schemes is described in detail in [4]. In the following subsections we suppose that an XSEM-ER schema and XSEM-H views are prepared. We describe how a conceptual level grounding can be specified and how the SAWSDL representation can be derived automatically.

4. CONCEPTUAL LEVEL GROUNDING

Firstly, we show how the derived ontology is derived from the XSEM-ER schema ((4) at Figure 4). We suppose OWL [10] as an ontology language. We represent each entity and relationship type from the XSEM-ER schema as an OWL class. If an entity type T1 is a determinant of a weak entity type or a participant of a relationship type T2 we add an OWL object property with the domain C1 and range C2 where C1 and C2 are the OWL classes representing T1 and T2, respectively. Attributes of entity and relationship types are represented by OWL data type properties. Even though we can not reconstruct the XSEM-ER schema back from the derived ontology (for example a class can represent an entity or relationship type) it contains sufficient information for the conceptual grounding.

Figure 5 shows a part of the derived ontology for the weak entity type Order from Figure 2 in the triple notation. The
names of classes in the derived ontology are given by the corresponding types in the XSEM-ER schema. For example, there is a class Order for the weak entity type Order. 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 hasCustomer with the domain Order and range Customer. Properties representing attributes are named in the same way. The rest of the ontology for the remaining entity and relationship types can be derived similarly.

As the conceptual grounding of the XSEM-ER schema to a target ontology the designer provides a mapping between the derived and target ontology ((5) at Figure 4). It means to map concepts from the derived ontology to corresponding concepts in the target ontology. For this we can profitably utilise existing languages for ontology mapping such as [5]. In [2] a survey of mapping languages is provided. We can also use OWL constructs for basic mapping. Therefore, we do not need to provide any extending constructs to ontology languages and we do not need to provide a new mapping language between XSEM-ER schemes and ontologies which simplifies our approach.

5. SAWSDL GROUNDING REPRESENTATION

The architecture shown at Figure 4 facilitates the design and management of structural descriptions of services and grounding to ontologies. However, it will also be required by the provider and clients to have the grounding represented with existing languages such as SAWSDL or WSMO. In this section we show how to derive SAWSDL representation of the conceptual grounding automatically. 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 loweringSchemaMapping are used to specify XSLT mappings between the structural and semantic representations. The derivation of SAWSDL representation of the grounding of an XML schema to a target ontology means to derive the XML schema from the corresponding hierarchical view and to enrich it with mappings of its components to the ontology.

Firstly, we show the derivation of XML schemes from hierarchical views. We use XML Schema language [6]. If a node

```xml
<!-- dont represents the derived ontology -->
<xs:element name="orderRequest" type="Order"/>
<xs:complexType name="Order"
  sawsdl:modelReference="dont#Order"
  sawsdl:liftingSchemaMapping="Order.xslt">
  <xs:element name="orderRequest" type="Order"/>
  <xs:element name="hasNumber"
    sawsdl:modelReference="dont#hasNumber"/>
  <xs:element name="hasAmount"
    sawsdl:modelReference="dont#hasAmount"/>
</xs:complexType>

<xs:element name="Product"
  sawsdl:modelReference="dont#Product">
  <xs:element name="number"
    sawsdl:modelReference="dont#number"/>
</xs:complexType>

<xs:element name="Item"
  sawsdl:modelReference="dont#Item">
</xs:complexType>
```

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 is given by the attributes of the node and the edges going from the node. Each attribute and edge going to a labeled child node is translated to an element declaration with the corresponding type. An edge going to a child node without a label is translated to a reference to the group representing the child node.

Figure 6 shows a part of an XML schema derived from OrderRequestView at Figure 3. There is an element declaration orderRequest for the root node from the view with the corresponding complex type definition Order. It contains element declarations corresponding to attributes shipDate and shipAddr 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 at the figure.

Secondly, we show the derivation of the grounding for XML schemes derived from hierarchical views to a target

```xml
(Order rdf:type owl:Class .
  :hasShipDate rdf:type owl:DataTypeProperty ;
    rdfs:domain :Order;
    rdfs:range xs:date .
  :hasShipAddr rdf:type owl:DataTypeProperty ;
    rdfs:domain :Order;
    rdfs:range xs:string .
  :hasCustomer rdf:type owl:ObjectProperty ;
    rdfs:domain :Order;
    rdfs:range :Customer .
  :hasItem rdf:type owl:ObjectProperty ;
    rdfs:domain :Order;
    rdfs:range :Item .

Figure 5: Derived Ontology
```
ontology. There are two possibilities. The simpler possibility is to derive the grounding of each XML schema to the derived ontology. Then we can use a semantic reasoner to dynamically translate between the semantic representations given by the derived and target ontologies using the specified mapping. The more complex but more practical possibility is to derive the direct grounding of an XML schema to each target ontology. However, we show only the first possibility because it is easier to describe. On the other hand, the second possibility uses the same idea as a base extended with a composition of the grounding to the derived ontology with the mapping of the derived ontology to a target ontology.

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. 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 hasShipDate and the element definition customer is bounded with hasCustomer. The resulting binding for our example is shown at Figure 6.

Secondly, the lifting and lowering XSLT mappings are generated automatically for the XML schema. We show how to generate the lifting schema (the lowering schema can be generated in a similar way) which transforms XML messages from the structural to semantic derived ontology representation. The semantic representation is serialised in RDF/XML. For each node in the hierarchical view we create an XSLT template matching the corresponding elements. This template transforms the matched elements to their semantic representation. We start with a template matching the root element orderRequest. According to the derived ontology, the element is transformed to its semantic representation serialised in RDF/XML as an element Order. The child elements of orderRequest, i.e. shipDate, shipAddr, customer, and item, correspond to the properties that are serialised to the child elements of the element Order in the resulting RDF/XML representation, i.e. hasShipDate, hasShipAddr, hasCustomer, and hasItem, 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 in the same way.

6. CONCLUSIONS

In this paper we showed how to use a conceptual model for binding structural and semantic descriptions of SWS which is called grounding. We also showed how to automatically translate the proposed conceptual grounding to the representation using existing technologies, concretely SAWSDL.

There are also limitations of the proposed approach. We discussed only data grounding, i.e. grounding of the structural description of the data. However, we can also describe the semantics of the operations provided by a web service in a target ontology and we can similarly provide a grounding of our operations to their semantic description in the target ontology. This possibility is not discussed in this paper and is a subject for our future research.

7. ACKNOWLEDGMENT

This paper was supported by the National programme of research (Information society project 1ET100300419) and by Grant Agency of Charles University (GAUK), grant number 204-10/257190.

8. REFERENCES