Grounding Web Services Semantically: Why and How?

Konstantin Pantschenko, Olaf Noppens, and Thorsten Liebig

University of Ulm, Dept. of Artificial Intelligence, Germany
{pantschenko|noppens|liebig}@informatik.uni-ulm.de

1 Introduction

Recent modeling frameworks for semantically enriched web services propose a layered architecture of different functionalities and representations. For discovery, selection, and composition of web services, an adequate *semantic representation* and a *transport layer* are (among others) necessary elements [1]. The semantic layer takes care of a shared understanding between requester and provider in order to enable automatic service provisioning. The most current approaches propose to utilize ontologies for this task [2]. The far end with respect to semantics is the transport layer, which specifies the details of how to access and communicate with the service implementation on a technical level. In order to invoke a service both layers need to be connected for data hand over between each other. This requires a mapping from an abstract semantical model to a concrete service specification in terms of a particular protocol, message format, data types, and serialization. We argue, that this bi-directional mapping, also called grounding, cannot be done syntactically as proposed by the OWL-S [3] service framework for example. This problem of linking declaratively specified parameter descriptions with primitive types (XML Schema in case of WSDL [4]) largely has been neglected in current semantic web services frameworks. In order not to limit the portability and practicability of current framework specifications we aim to bring this issue to the fore. In the following we will describe the mismatch of syntax-based groundings and propose a semantic mapping framework for which we refer to an existing algorithm and shortly describe our mapping editor

2 A Semantic Mapping Approach for Grounding Web Services

Common to all semantic web service frameworks is their distinction between a semantical layer for description of a services characteristics and an execution layer for concrete service invocation. In order to facilitate the execution of a service a so called grounding is needed for mapping the semantical space into the physical data space at transport level. The semantical layer is based on ontological modeling principles. An encoding of such an ontological model may result in many syntactically different serializations caused by encoding alternatives of

the language elements. Therefore, a syntactical mapping, like XSLT as proposed by the OWL-S specification, is not adequate here.

In case of grounding an OWL-S service this issue comes into play when linking parameters of OWL-S process steps with WSDL messages consisting of XML Schema values. As can be seen on the left hand side of figure 1, an output mapping from a XML Schema definition to OWL (from a lower to a higher level description) is less problematical. In contrast, an input mapping, namely a

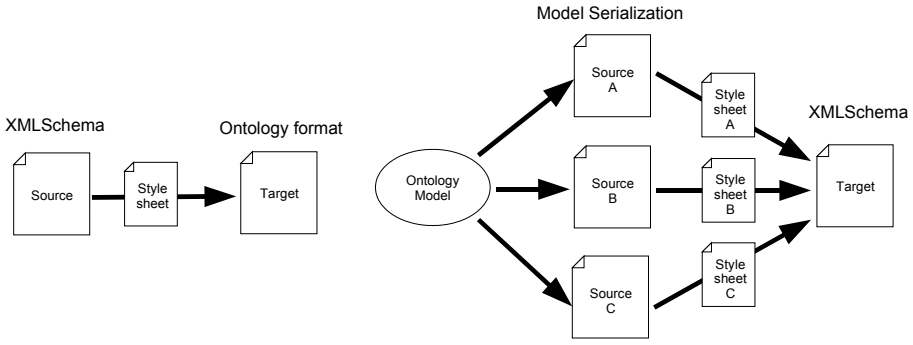

Fig. 1. XSL Transformation problem

mapping from OWL to XML Schema, may result in different serializations (see right hand side of figure 1). In oder to guarantee general applicability this kind of mapping has to provide a style sheet for every possible serialization. Even worse, the existence of nested relations or references results in an exponentially growing number of serializations.

3 A Mapping Approach for Grounding Web Services

As described in the previous section a syntactical transformation between abstract ontology based parameter and their corresponding low-level data types for a concrete service grounding is not sufficient. The following shortly describes our semantic transformation approach [5]. Our approach is neither limited to a specific semantic service framework (like OWL-S or WSMO) nor to a specific Web Service description language like WSDL. Only for demonstration purpose we will refer to OWL-S as the service ontology language and to WSDL as web service description language in the following.

A mapping establishes a bidirectional link between OWL types and XSD type definitions for a specific Web service. The mapping will translate input parameter values from OWL instances to XSD types as well as the other way around, e. g. after a successful execution back to ontology instances for further processing. Such a mapping can easily be specified with the help of a declarative transformation description represented as an RDF model. Note that from an

OWL perspective only filler of the `owl:DatatypeProperty` can be transformed to an instance of an `xsd:simpleType`. Moreover, `xsd:complexType`s are decomposed into simple types and therefore to fillers of a datatype property.

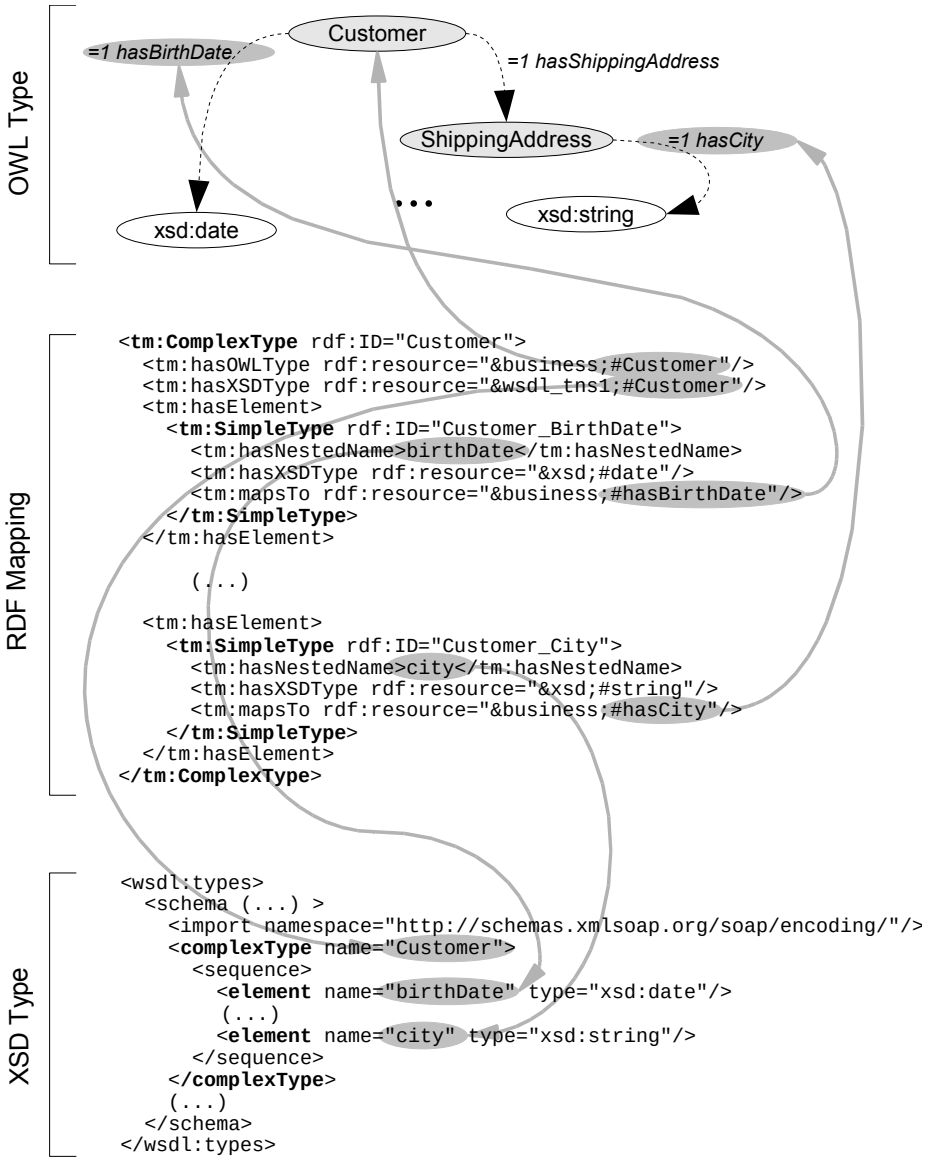

Fig. 2. An example mapping schematically.

Figure 2 shows an example mapping. Here, the OWL class 'Customer' is mapped to the 'Customer' type of the concrete service description. Note that `tm:hasXSDType` tags refer to XSD datatypes in the WSDL document whereas `tm:mapsTo` refers to the corresponding datatype property in OWL. XSD types of the service description always have a flat structure, so all XSD types are aggregated in one complex type as a sequence of simple type elements (denoted by `tm:hasElement`). For example, the `tm:SimpleType` 'Birthday' maps a filler of the datatype property 'hasBirthDate' to an value instance of 'xsd:date'. Moreover, the tag `tm:hasNestedName` refers to the corresponding attribute of the complex type 'Customer' of the service description. Our mapping approach is not sensitive to syntactical serializations because all mappings are declaratively specified on a semantical level within a RDF model.

A detailed description of an algorithm which actually performs the mapping and discussion about related work can be found in [5].

4 The Type-Mapping Workbench

We have implemented an editor which guides and assists the user (i.e. service provider) through all the steps needed to create a mapping as described in the previous section. Note that currently, it only supports the mapping between WSDL datatypes and OWL types.

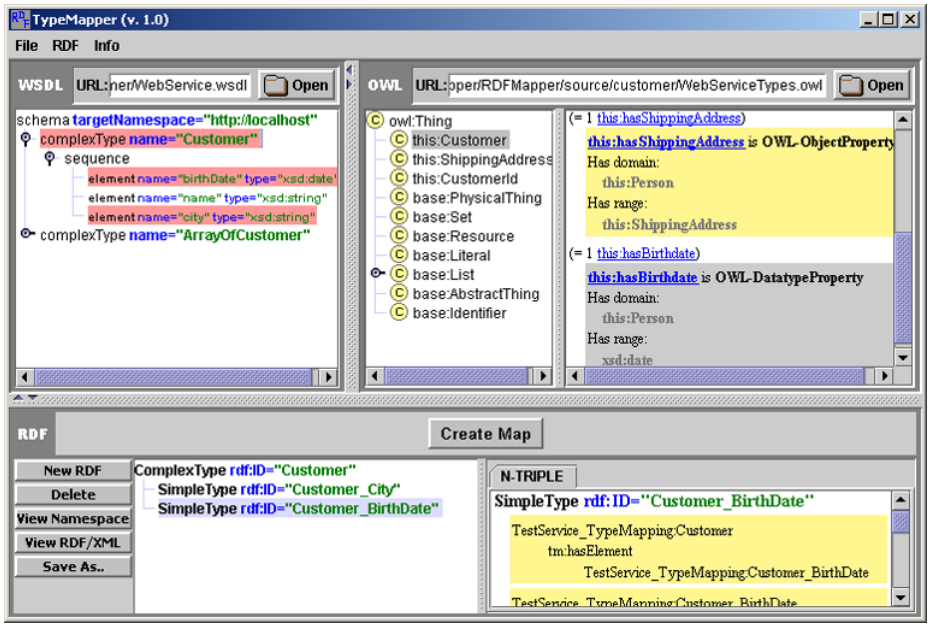

Fig. 3. Type-mapping editor with WSDL in upper left, OWL ontology in upper right and a mapping structure in the bottom compartment.

Figure 3 shows the user interface which is divided into three panes. On the left hand side, all type information (e.g. complex types and simple types) of a WSDL document are displayed. After loading the corresponding OWL ontology document, all class types and restrictions on properties are shown in a taxonomy tree on the right hand side. Next, the user creates a new (empty) mapping which will be displayed at the bottom of the display area. To establish a new map between a WSDL complex type and an OWL datatype, the user needs to select the corresponding elements in the WSDL and the OWL view, and then click on 'CreateMap' to create a new mapping structure. The editor semi-automatically creates the corresponding structure or offers a list of choices for further specification of the type of mapping. Possible types are `ComplexType`, `SimpleType` or `ArrayType`. For instance, to build a mapping as shown in figure 2, one first selects the entry for the complex type 'Customer' in the WSDL display and the OWL class 'Customer' which results in the creation of a mapping structure 'ComplexType Customer' in the mapping display. After selecting the simple type 'birthday' (WSDL entry) and the datatype property 'hasBirthday' of cardinality restriction '= 1 hasBirthday', the user can create the corresponding mapping structure (by clicking on the button 'CreateMap'). Restrictions are displayed on the right display area next to the selected OWL class. In a similar way all other map entries of figure 2 can be created. Moreover, the building of a complex type mapping structure may involve to exploit a chain of OWL classes. In our running example of figure 2, to create the mapping structure for the WSDL attribute 'City' the user may select 'Customer', the range of property 'hasStreet' which lead one to the OWL class 'ShippingAddress' and to the property 'hasCity'. For an `ArrayType` it is necessary to choose one of the already created types. At any stage, it is possible to remove map entries and to view or save the mapping as an RDF model in XML format. Already linked WSDL elements are marked with a background color and are not longer selectable unless the user removes the corresponding map entry.

References

1. Bussler, C.: B2B Protocol Standards and their Role in Semantic B2B Integration Engines. IEEE Data Engineering **24** (2001)
2. Cabral, L., Domingue, J., Motta, E., Payne, T., Hakimpour, F.: Approaches to Semantic Web Services: An Overview and Comparisons. In: First European Semantic Web Symposium (ESWS2004), Heraklion, Crete, Greece (2004)
3. Ankolekar, A.: OWL-S: Semantic Markup for Web Services (2003) `http://www.daml.org/services/owl-s/1.0/owl-s.pdf`.
4. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1. Technical report, Word Wide Web Consortium (2001) `http://www.w3.org/TR/2001/NOTE-wsdl-20010315`.
5. Balzer, S., Liebig, T.: Bridging the Gap Between Abstract and Concrete Services – A Semantic Approach for Grounding OWL-S –. In: Proceedings of the Workshop on Semantic Web Services: Preparing to Meet the World of Business Applications, Hiroshima, Japan (2004) 16–30