Towards Translating between XML and WSML based on mappings between XML Schema and an equivalent WSMO Ontology

Matthew Moran, Adrian Mocan

Digital Enterprise Research Institute, DERI, Ireland
{matthew.moran, adrian.mocan}@deri.org

Abstract. It is very important that any infrastructure for handling Semantic Web services be compatible with existing Web services whose description is already available through a corresponding Web service Description Language (WSDL) [8] document. In the case of the Web Services Execution Environment (WSMX) [9], only Web services having a valid semantic description expressed in terms of the Web Services Modeling Ontology (WSMO) [11] and represented using the Web Services Modeling Language (WSML) [10] will be available for discovery, invocation or composition. This leads to the question of how to deal with Web services without such a description. There are two aspects to consider. The first is that the data model for WSDL services is defined in terms of XML Schema while the data model for Semantic Web services, in the context of WSMX, is defined conceptually by one or more WSMO ontologies and is expressed in WSML. The second aspect is how to map between the description of the public interface(s) and their physical bindings offered by a Web service as defined in WSDL and the corresponding descriptions offered by the description of the public behaviour of a Semantic Web service described in WSML. This paper describes early work in dealing with the first aspect mentioned.

1. Introduction

Semantic Web services aim to improve the possibilities for automated discovery, composition and invocation of Web services by providing ontology-based service descriptions expressed in a formal language with support for reasoning. In terms of adoption for critical enterprise tasks, Web services are still at an early stage. However the technology is well understood and there is growing momentum for using Web services at the core of service-oriented systems. With this in mind, if the technology infrastructure for Semantic Web services is to be successful, it is very important that this infrastructure be able to communicate with existing Web services described with WSDL. In the context of WSMX, only Web services with a semantic description expressed in terms of WSMO are available for the operations of discovery, composition or invocation. Therefore, the first step in making a service having only a WSDL description available to WSMX is to create the corresponding WSMO description. This makes the service *description* available to WSMX but does not solve the problem of how the *implementation* of the service can be invoked. There are two aspects to this problem. The first is that the data model of the input and output messages for WSDL services is defined using one or more XML Schema while the data model for a WSMO service is defined using the conceptual model provided by one or more WSMO ontologies. This leads to a requirement for a mapping between the XML Schema of a WSDL service description and the ontology(ies) used by the corresponding WSMO service description. The second aspect to the problem is how the description of the public interface of the Web service as defined in WSDL can be mapped to the corresponding part, called choreography (defined in [12]), of the WSMO description. This paper only deals with describing initial work on addressing the first aspect.

Figure 1 shows a high level view of steps proposed above for making a WSMO service description available to WSMX. These steps are part of a high level proposal and *not* part of the present WSMO definition of a Web service. Step 1 indicates the creation (with tool support) of a WSMO description for an existing Web service. Step 2 is the creation of an ad-hoc ontology that is equivalent to the XML Schema used by the WSDL description. This step is intended to be automatic based on mappings between the conceptual model for WSMO ontologies and the conceptual model for XML Schema. Step 3 is the creation of mapping rules based on the mappings mentioned for step 2. Finally, step 4 is intended to suggest that the mapping rules should be stored as part of the service description.

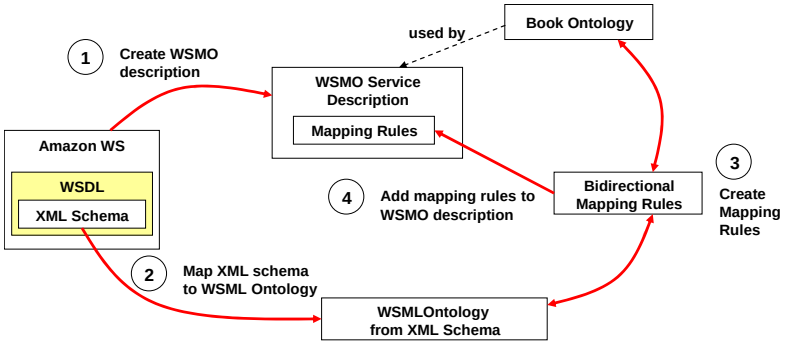

Figure 1: Overview – making a WSDL service available to WSMX

In this paper we focus on the problem of how to translate between data instances expressed in XML and data instances expressed in WSML. Our approach is based on creating mappings at the conceptual level between the XML Schema used by the WSDL service description and the WSMO ontology used by the WSMO service description. We use the term *lowering* to describe the translation of data instances represented in WSML to XML and the term *lifting* to describe the translation of data instances from XML to WSML. The work is at an early stage and while it seems to be an attractive approach to the problem, it is not yet certain that it will prove to be practicable. As will be seen in the related work, handling the relationship between XML Schema and ontologies is not a new problem and we aim to build on the work already carried out by others in this area.

The remainder of the paper is organized as follows. Section 2 provides some information on the authors. Section 2 reviews related work. Section 3 presents the main topic of the paper. Section 4 provides a discussion of some of the issues identified in the work to date and section 5 presents a conclusion and some anticipated next steps.

2. Related Work

There has been significant work done on investigating the relationship between ontologies and XML Schema and whether or not it is relevant to compare the two. In [1], the authors demonstrate that although ontologies and XML Schema provide data description at different levels of abstraction, it is useful to compare the languages used to represent the data, and to derive some conclusions from the similarities and differences uncovered. Using the Ontology Inference Layer (OIL), defined in [2], as the ontology language for the comparison, they highlight the contrasting purposes of both languages as the core difference between them. XML Schema is concerned with defining the vocabulary and the constraints on the structure of well-formed XML documents. Ontology languages, including OIL and WSML, are concerned with representing the formal specification of a shared domain theory. Ontologies are not explicitly concerned with defining the structure of documents or even the exact representation of data items (e.g. format of a date datatype) but rather defining the meaning of the data that the documents contain. Significantly, the authors of [1] outline as the results of their comparison that both XML Schema and OIL can each be more expressive depending on the viewpoint adopted when defining the data model. The authors also propose a methodology for translating an ontology specification (in this case, OIL) into an XML Schema. Our approach for the translation between WSML and XML is intended to be based on those presented in [1]; with the difference that we are focusing on lowering ontology instances to XML instances and are not interested in obtaining an XML Schema from ontology.

Other attempts in this direction, have attempted to add semantic meta-information to instance data, to make it available for tasks involving reasoning. Such an approach is proposed by [3] which develops an RDF interpretation for XML documents. It considers that both structural and semantic mark-up can coexist in the same document and uses RDF to annotate existing XML documents. Another type of approach is the one that does not require changes to the involved technologies. In [4] the authors propose sets of mappings, the first one used for lifting XML Schemas to OWL ontologies and the second one to translate XML documents into RDF graphs. Our approach takes as model for the lifting operations this last approach, with

the difference that only one set of mappings is created (between XML Schema elements and WSMO ontology meta-model) and used for both lifting operations: schema lifting and instance lifting as well.

The symmetric part of the problem, the lowering, was addressed in several other works investigating the relations between ontology representation languages and document structuring techniques (schemas) [1]. In [5] it is shown how an XML document type definition (DTD) is generated from a given ontology. By this, XML instances are linked to that ontology and it makes possible the intelligent information integration and retrieval from the semi-structured documents (i.e. XML documents).

3. Translating between XML and WSML based on Conceptual Mappings

This work stems from the requirement for WSMX to be able to communicate with Web services where the data sent and received by those services is described by a XML Schema included as part of the service's WSDL description. As WSMX sends and receives WSML messages, we require translation in both directions for communication to be possible:

- from WSML instances to XML instances (lowering)
- from XML instances to WSML instances (lifting)

There are at least two approaches that can be taken. The first approach and focus of this paper, proposes *lifting* the XML to an ontological representation in WSML so as to take advantage of the results coming from research into data mediation. Data mediation tackles the problems of how to bridge between two data models that may have differences in terms of semantics and syntax. This is a significant issue for Web service communication where the server requester and provider are often heterogeneous and autonomous entities with independent data models. Thus, the problems faced by WSMX in communicating with WSDL Web services is not just a question of differing data syntax but, more significantly, a question of differing data semantics. Reusing emerging data mediation tools avoids reinventing a solution for this problem.

The second approach, not described in detail here depends on using the XML syntax for WSML. XSLT is used to directly translate between the WSML ontology instances written in XML syntax and the XML instances corresponding to the XML Schema defined by the WSDL for the service. This approach is outside the scope of this paper.

Overview of Approach

Figure 2 shows three layers of abstraction for both the XML and WSMO approaches. At the top of the stack is the meta-model layer. For XML, this is provided by the XML Schema Conceptual Framework [6]. For WSMO this is provided by the Ontology metamodel definition.

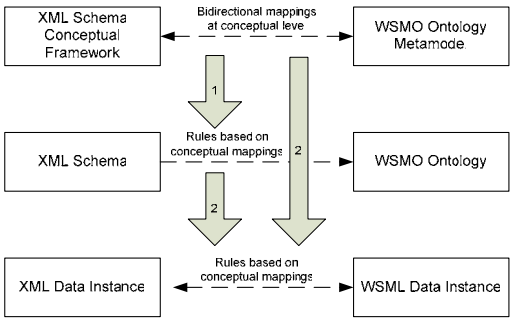

Figure 2: Layers of Abstraction for WSMO and XML

The second level describes the model layer. For XML, these are instances of XML Schema documents. In WSMO this is represented by instances of WSMO Ontologies. Finally the bottom layer for XML is instances of XML data and for WSMO, it is instances of ontological concepts represented in WSML.

The fundamental task is to create bidirectional mappings between the XML Schema Conceptual Framework and the WSMO Ontology Metamodel. Once created, these mappings should be used to automatically enable the creation of a WSMO ontology for any particular XML Schema. The resulting ontologies will be ad-hoc, but should nevertheless provide the gateway for re-use of the data mediation techniques being developed for WSMO. This is a key advantage compared to approaches that use XSLT to map between data sources based only on syntactic rules. It is worth noting that the focus of this work is on the XML Schema used in WSDL documents and that, in particular, the data models of these schemas are often quite small.

The bidirectional mappings at the conceptual level will also be used to create the mapping rules to transform XML data instances to WSML data instances and vice versa (large arrow, numbered 2). The creation of the mapping rules for translation between XML and WSML instances also requires input from the mappings used to create the ad-hoc ontology from the XML Schema (smaller arrow, numbered 2). In particular, this arises because of the greater expressiveness of XML Schema in prescribing the *structure* of an XML document. Instance data represented in WSML contains no structural information. This could lead to a problem, if the WSML instance data needs to be translated to an XML document whose structure must be validated against an XML Schema. This is discussed further in section 5.

Three distinct activities are required:

- The definition of a mapping from the XML Schema Conceptual Model to the WSMO Ontology metamodel.
- Application of this mapping to automatically create an ad-hoc WSMO ontology from a specific XML schema. This would be required each time a Web service whose data types are defined by an XML Schema in its WSDL document needs to be compiled to a WSMO compliant Semantic Web service framework such as WSMX.
- The third activity is the creation of bidirectional mapping rules to be used for the transformation between XML instances and WSML instances. The mapping rules are created once for each XML schema/generated WSMO ontology pair and are applied at run time for each data instance.

Short Overview of XML Schema

XML Schema provides a way to define constraints on the syntax and structure of an XML document. The specification for the XML Schema is substantial. We take an incremental approach to the development of a mapping between XML Schema and the WSMO Ontology metamodel by focusing on a subset of commonly used type and element definitions used by XML Schema. The authors of [1] provide a concise overview of the treatment of datatypes and structures of XML Schema and there is no point repeating this here. For the purpose of this document we look at how a subset of schema components map to components in the WSMO ontology metamodel. These components are: *simple type definition*, *complex type definitions*, *attribute declarations*, and *element declarations*. They are briefly described in the following section and examples are provided in Listing 1.

Simple Type Definition: XML Schema provides a wide range of built-in datatypes, for example, string, integer, boolean, float, decimal etc. These are examples of one form of simple type definitions. Another form of simple type definitions can be used to provide constraints on the values of built-in types. For example, it may be necessary to restrict the allowed values of the positiveInteger datatype to a particular maximum value. An example of this is included in Listing 1.

Complex Type Definition: Can be used to (i) define a datatype composed of sub-elements of other datatypes, (ii) define the allowed structure of child elements using the keywords *all*, *sequence* and *choice* or (iii) extend or restrict the definition of an exiting complex type. Additionally, the values of elements can be accompanied by constraints on their values. Listing 1 provides an example of this.

Attribute Declarations: Attributes can be either global or associated with a particular complex type definition. Attributes are an association between a name and a simple type definition.

Element Declarations: An element declaration is an association between a name and a type definition, either simple or complex. The scope of element declarations can be either global or in the scope of a containing complex type. Elements declarations define what element instances are valid for an XML document conforming to that XML Schema.

We use the example of a simple fragment of an XML Schema (modified from an example in [7]) in Listing 1 to illustrate our approach to these steps.

```
<xsd:schema xmlns:xsd="http://www.ws.org/2000/08/XMLSchema">
<xsd:element name="resumes" type="resumeTypes"/>
<xsd:complexType name="resumesTypes">
   <xsd:sequence>
       <xsd:element name="applicantName" type="xsd:string">
       <xsd:element name="jobsAvailable" type="jobListType">
   </xsd:sequence>
   <xsd:attribute name="applicationDate" type="xsd:date"/>
</xsd:complexType>
<xsd:complexType name="jobListType">
   <xsd:sequence>
       <xsd:complexType name="job" type="jobDesc">
           <xsd:attribute name="jobid" type="xsd:string"/>
       </xsd:complexType>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="jobDesc">
   <xsd:element name="title" type="xsd:string">
   <xsd:element name="salary">
       <xsd:simpleType>
           <xsd:restriction base="xsd:positiveInteger">
               <xsd:maxExclusive value="55000">
           </xsd:restriction>
       </xsd:simpleType>
   </xsd:element>
</xsd complexType>
</xsd schema>
```

Listing 1: Example of XML Schema Definition

Identifying Mappings from XML Schema to WSML

We propose initial simple mappings for each of the components described in the last section and, based on this, provide a sample equivalent WSMO ontology fragments expressed in WSML. We emphasize that this is early work and its scope only covers a subset of the possibilities for XML Schema that allow us to develop an initial strategy for creating a more comprehensive mapping. That said, in the course of identifying the mappings described here, it became clear that the structural information present in the XML Schema is not automatically reflected in the WSMO ontology. We return to this point in the discussion section at the end of the paper.

Simple Type Definitions. For the built-in types, no mapping is required as WSML supports the use of the built-in types defined by the XML Schema namespace at `http://www.ws.org/2000/08/XMLSchema`. Where a simple type definition is used to create a new type based on restricting a built-in type, we propose the creation of a new subconcept of the built-in type in WSML along with an axiom to define the

restriction. For example, the simple type definition in Listing 1 used to define the type for salary elements would map to the following WSMO concept definition:

```
concept salaryType subConceptOf xsd:positiveInteger
     constraint salaryConstraint
          definedBy ?X:salaryType and ?X<55000
```

Listing 2: Result of Mapping Simple Type Definitions

Complex Type Definitions. Complex type definitions can contain sub-components that are a mixture of elements, attributes and other complex type definitions. As described above they can also contain keywords to indicate the correct structure for XML to be compliant with the type (*sequence*, *all* and *choice*). Complex types always map to a concept in WSMO. Sub components with a simple built-in type are mapped to attributes of the concept with the same built-in type. Sub components with simple types that are not built-in are mapped to attributes with the type of the mapped simple type definition. A sub component that itself is a complex type leads first to the creation of a corresponding concept. Then the sub component is mapped to an attribute with the type of the newly created concept.

Attributes within the scope of complex type definitions provide an additional complexity. In terms of WSML, we propose that the attribute acts as an extension to the definition of the concept corresponding to the type of the complex type definition. This means that (i) a concept for the complex type is created, (ii) a second concept is defined as a subconcept of the concept created in (i). This second concept contains as attribute the XML Schema attribute definition. The following WSMO definitions are the results of the mappings from the complex types contained in Listing 1. In particular, the mapping of the complex type definition with the name attribute, job, illustrates the handling of the attribute.

```
concept resumeTypes
     applicantName ofType xsd:string
     jobsAvailable ofType jobListType
concept jobListType
     job (0 *) ofType jobList_withAttrJobID
concept jobList_withAttrJobID subConceptOf jobDesc
     jobid ofType xsd:integer
concept jobDesc
     title ofType xsd:string
     salary ofType salaryType
```

Listing 3: Result of mapping Complex Type Definitions

Attributes. The handling of attributes within the scope of the definition of a complex type was shown above. We have not yet investigated the handling of attributes in other situations.

Elements. Elements are structural components within an XML document. There is no corresponding component in a WSML ontology.

4. Discussion Points

As stated in [1] there are many differences between ontologies and XML Schema which from one viewpoint suggests that it is not a good idea to identify a mapping between them. Ontologies in many ways provide a higher level of abstraction than XML Schema and a more expressive way to deal with the specification of a conceptual model including representing relationships between concepts and the formal axioms constraining on how the concepts can be instantiated. On the other hand XML Schema provide a way to define both the *vocabulary* and *structure* that a compliant XML document must use. WSML is less verbose than XML but it depends on the reader's perspective which of the two languages is more human-readable. What is certain is that XML is a purely syntactic language and although the use of XSLT to transform XML documents to other formats is quite powerful, it is still rooted in syntax and takes no

account of the semantics of the underlying data. Using a combination of XML and XSLT to transform between different schema definitions at the data instance level provides little scope for reuse. We believe that by developing a mechanism to lift XML instances to equivalent WSMO instances based on mapping between the XML Schema and the WSMO Ontology offers a more flexible approach. We also believe that the potential for reuse of data mediation tools is high and that much of this process can be automated.

One of the problems that became evident even at this early stage of the work is the significance of losing structural information about the XML document during translation to WSML. For example, when mapping an instance of XML conforming to the XML Schema to an instance of a WSMO ontology fragment conforming to the created WSML ontology, the information regarding which WSML elements corresponded to XML Schema elements and which to XML Schema attributes was lost. The name of the elements was also lost as elements, being structural components, were not included as part of the mapping. This would cause obvious problems when translating from a WSMO instance to an XML instance conformant to an XML Schema. We need to identify how the mapping can be extended to maintain the minimum required structural information.

As this is work-in-progress, we have only commented on a small part of the overall problem. In this paper, we do not describe the mappings from WSMO to XML Schema or how the mapping rules to actually translate between instances of XML and WSMO can be designed and implemented. This would be a welcome topic for discussion at the workshop. In the introduction, we named two aspects of making WSDL services available to WSMX. The first has been the topic of this paper. The second is how the interface and message exchange patterns for a Web service described in WSDL can be mapped to the corresponding choreography description for a WSMO Semantic Web service. This would also be an interesting discussion for the workshop.

5. Conclusions and Future Work

This short paper presents some initial work in creating a bidirectional mapping between XML Schema and the WSMO Ontologies expressed in WSML. Such a mapping should enable the automatic creation of WSMO ontologies on-the-fly from XML Schema and facilitate the re-use of data mediation tools already designed and implemented to create mappings between the generated ontology and other external domain ontologies. We presented the motivation for this work as the enabling of communication between the WSMX environment for Semantic Web services and existing Web services whose data-types are described using XML Schema embedded in WSDL.

We reviewed related work that has been carried out already in defining the relationship between ontologies and XML Schema and identified the work of [1] as a very useful starting point for this research. After describing an overview of our approach, we described how a subset of the XML Schema language can be mapped to WSMO using an example to help illustrate the mapping. We make clear that the choice of the subset was used to have an easily manageable set of language components from XML Schema to develop a prototype of the mapping and to get an early feeling for challenges and problems.

There is significant future work to be carried out to demonstrate that this approach is practicable but early results are promising. The next steps are to continue developing the bidirectional mappings between XML Schema and the WSMO Ontology, and to develop an algorithm based on those mappings for the creation of a WSMO ontology from an XML Schema definition.

Acknowledgements

The work is funded by the European Commission under the projects DIP and ASG and by Science Foundation Ireland (SFI) under the DERI-Líon project.

References

1. Klein, M., Fensel, D., van Harmelen, F., Horrocks, I.: The Relation between Ontologies and XML Schemas. Linkoping Electr. Art. in Comp. and Inf. Sci. 6 (2001)
2. Horrocks, I., Fensel, D., Broekstra, J., Decker, S., Erdmann, M., Goble, C., Van Harmelen, F., Klein, M., Staab, S., and Studer, R.: OIL: The Ontology Inference Layer, Technical Report IR-479, Vrije Universiteit Amsterdam, September 2000.
3. Melnik, S.: Bridging the Gap between RDF and XML, http://www-db.stanford.edu/~melnik/rdf/fusion.html
4. Trastour, D., Ferdinand, M., Zirpins, C.: Pragmatic Reasoning- Support for Web-Engineering: Lifting XMLSchema to OWL, ICWE 2004, Munich, Germany.
5. Erdmann, M., and Studer, R.: Ontologies as Conceptual Models for XML Documents. Twelfth Workshop on Knowledge Acquisition, Modelling and Management, KAW 1999, Alberta, Canada
6. XML Schema Part 1: Structures Second Edition, W3C Recommendation, 28 October 2004. http://www.w3.org/TR/xmlschema-1/#concepts
7. Griffen, G.: XML and SQL Server 2000, p32, New Riders Publishing, 2002, ISBN 0-7357-1112-7
8. Chinnici, R., Gudgin, M., Moreaum, J.J., Weerawarana, S., and Schlimmer, J.. Web services Description Language (WSDL) Version 1.2. World Wide Web Consortium, March 2003. Available from http://www.w3.org/TR/2004/WD-wsdl20-20040326/
9. Cimpian E., Moran, M., Oren, E., Vitvar, T., Zaremba, M.: Overview and Scope of WSMX. WSMO Working Draft v0.2, 08 February 2005
10. de Bruijn, J., Lausen, H., Krummenacher, R., Polleres, A., Fensel, D.: The WSML Family of Representation Languages. WSMO Working Draft, 24 December 2004, available at: http://www.wsmo.org/2004/d16/d16.1/v0.2/20041224/
11. Roman, D., Lausen, H. and Keller, U. (eds.): Web service Modeling Ontology - Standard (WSMO - Standard), WSMO deliverable D2 version 1.0. available from http://www.wsmo.org/2004/d2/v1.0/.
12. Roman, D., Scicluna, J., Feier, C., Stollberg, M., and Fensel, D.: Ontology-based Choreography and Orchestration of WSMO Services, WSMO Deliverable D14.02, v 0.2 available at http://www.wsmo.org/TR/d14/v0.2/20050301/