Semantic Web Services Grounding *

Jacek Kopecký[1], Dumitru Roman[1], Matthew Moran[2], and Dieter Fensel[1,2]

[1] Digital Enterprise Research Institute, Innsbruck
[2] Digital Enterprise Research Institute, Galway
{firstname.lastname}@deri.org

Abstract

Semantic Web Services frameworks like OWL-S and WSMO combine semantic descriptions of Web service capabilities, inputs, outputs and behavior with the syntactic interface descriptions in WSDL and XML Schema. The glue between the semantic and syntactic description layers is called grounding. In this paper we identify the uses for grounding and we present the existing grounding approaches and propose new ones, discussing their respective advantages and drawbacks. Finally, we compare OWL-S and WSMO with regards to their support of the presented grounding approaches and we recommend which approaches should be considered for future work.

1. Introduction

Web services [8] are a relatively new, but maturing set of technologies for distributed computing, reusing much of the infrastructure of the World Wide Web.

Following the vision of Semantic Web [10] that aims to make the World Wide Web machine-readable and understandable, researchers are proposing Semantic Web Services (SWS), extending Web service technologies with semantics in order to automate many tasks common when using Web services. These tasks include, for example, discovery, negotiation, composition and invocation of services.

Automation of these tasks requires semantic description of various aspects of Web services, and two major SWS description frameworks have been proposed: WSMO [16] and OWL-S [18]. For example, the process of Web service discovery can be automated using a machine-processable description of what the user wants (a user goal) and what the available services can do (service capabilities). We call this kind of information *semantic description* of Web services.

*This material is based upon works supported by the EU funding under the projects DIP (FP6 – 507483) and ASG (FP6 – 004617), and by the Science Foundation Ireland under Grant No. SFI/02/CE1/I131.

However, currently deployed Web services are generally described only on the level of syntax, specifying the structure of the messages that a service can accept or produce. In particular, Web Service Description Language [6] describes a Web service interface as a set of operations where an operation is only a sequence of messages whose contents are constrained with XML Schema [5]. We call this the *syntactic description* of Web services.

Certain tasks require that semantic processors have access to the information in the syntactic descriptions, for example to invoke a discovered service, the client processor needs to know how to serialize the request message. Linking between the semantic and the syntactic description levels is commonly called *grounding*.

In section 2 below, we analyze why and when grounding is necessary and what pieces of Semantic Web Service descriptions need to be grounded; and we also discuss the general issues of Semantic Web Service grounding. We identify two major types of grounding, so in section 3 we talk about data grounding and in section 4 we talk about grounding behavior descriptions. In both sections, we list existing approaches to grounding and propose new ones; and we analyze WSMO and OWL-S groundings with regard to which approaches they support. In the end, we present our conclusions and future plans in section 5.

2. General grounding uses and issues

Semantic Web Services are described in terms of their functional and behavioral properties, using logics-based (ontological) formalism. First, to enable Web service discovery and composition, SWS frameworks need to describe *what* Web services do, i.e. service *capabilities*. Second, to make it possible for clients to determine *how* to communicate with discovered services, the *interfaces* of the services need to be described.

The description of a service interface must be sufficient for a client to know how to communicate successfully with the Web service; in particular a service interface must de-

scribe the messages and the networking details. For interoperability with existing Web services and infrastructures, interface description is based on WSDL. Grounding is the glue between the semantic interface description and WSDL.

As already mentioned, WSDL models a service interface as a set of operations representing message exchanges. Message contents are specified abstractly as XML Schema element declarations, and then WSDL provides so-called *binding* information with the specific serialization and networking details necessary for the messages to be transmitted between the client and the service.

On the data level, Semantic Web Service frameworks model Web services as entities that exchange semantic (ontological) data. Grounding must be able to transform the semantic data to XML messages that will be sent over the network (according to serialization details from the WSDL binding), and grounding must also specify how received XML messages are interpreted as semantic data. We investigate this aspect of grounding below in section 3.

A Web service interface in WSDL contains a number of operations. Within an operation, message ordering is prescribed by the message exchange pattern followed by the operation. WSDL does not specify any ordering or dependencies between operations, so to automate Web service invocation, a Semantic Web Service interface must tell the client what particular operations it can invoke at a specific point in the interaction. We call this the *choreography model*, and very different choreography models are used by the known SWS frameworks. However, since they all ground to WSDL, the grounding must tie the choreography model with WSDL's simple model of separate operations. This aspect of grounding is further detailed in section 4.

We now know what kind of information must be specified in grounding, and we have to choose where to place that information, assuming that the semantic description is in a document separate from the WSDL. There are three options for placing grounding information: 1) putting grounding in the semantic description document, 2) embedding grounding within WSDL, and 3) externalizing grounding in a third document.

Putting grounding within the semantic description format makes it straightforward to access the grounding information from the semantic data, which follows the chronological order of SWS framework tasks — discovery only uses the semantic data, and then invoking the discovered service needs grounding. This approach is currently taken by both WSMO and OWL-S.

On the other hand, putting grounding information directly in WSDL documents (option 2) can enable discovering semantic descriptions in WSDL repositories, for example in UDDI [3]. This approach is taken by WSDL-S [7], a specification of a small set of simple WSDL hooks that can be used with any SWS modeling framework. The rest of this paper considers WSDL-S alongside with OWL-S and WSMO, even though WSDL-S itself does not provide actual SWS modeling capabilities.

An externalized grounding (outside both WSDL and the semantic descriptions) does not provide easy access to the grounding information from either the semantic or syntactic side, but it may introduce even more flexibility for reuse. Such externalized grounding is, however, not supported by any current specification.

The options listed above are not exclusive, as grounding information can be put redundantly both in the semantic description document and in the WSDL, for example, so that it is available from both sides. This could be done by using the native grounding mechanism in WSMO to point to WSDL and at the same time annotating the WSDL with WSDL-S elements pointing back to WSMO.

3. Data grounding

Web services generally communicate with their clients using XML messages described with XML Schema. On the semantic level, however, Web service inputs and outputs are described using ontologies. A semantic client then needs grounding information that describes how the semantic data should be written in an XML form that can be sent to the service, and how XML data coming back from the service can be interpreted semantically by the client. In other words, the outgoing data must be transformed from an ontological form to XML and, conversely, the incoming data must be transformed from XML to an ontological form.

Since the semantics of XML data is only implicit, at best described in plain text, a human designer is required to specify these data transformations so that they can be executed when a semantic client needs to communicate with a syntactic Web service.

In figure 1 we propose a way of distinguishing between data grounding approaches. The figure shows a fragment of the ontology of the semantic description of an example Web service in the upper right corner and the XML data described in WSDL in the lower left corner. The three different paths between the XML data quadrant and the semantic quadrant present different options where the transformations can be implemented: on the XML level, on the semantic level, and a direct option spanning the two levels. (Note that we use WSMO terms in the figure but it applies equally to OWL ontologies.)

First, since Semantic Web ontologies can be serialized in XML, an XSLT (or similar) transformation can be created between the XML data and the XML serialization of the ontological data. This option is detailed below in section 3.1. In case the XML format of the ontological data is actually suitable for a particular Web service, the transformation can be avoided; this case is discussed separately in section 3.2.

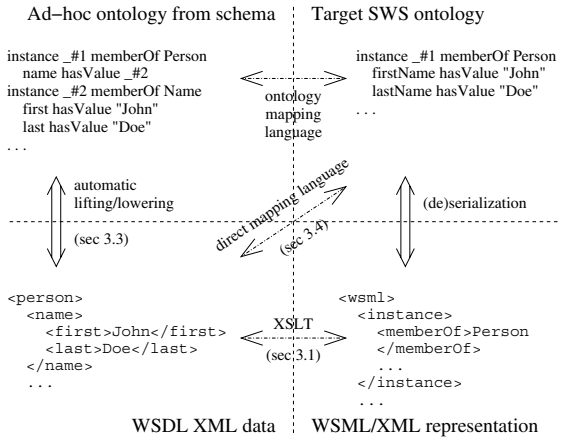

Figure 1. Data grounding approaches

Second, an ad-hoc ontology can be generated from the XML Schema present in the WSDL document, with automatic transformations (*lifting/lowering*) between the XML data and their equivalent in the ad-hoc ontology. Then a transformation using an ontology mapping language can be designed to get to the target ontology. This approach is detailed in section 3.3, together with the special case when the generated ad-hoc ontology is in fact suitable for the semantic description of the given Web service and no manual transformation is needed.

Finally, a direct approach for mapping between XML data and the target semantic data is described in section 3.4.

3.1. XML-level transformations

Both major SWS frameworks have an XML format for their ontological data: OWL-S uses OWL as the ontology, so any data can be represented in RDF/XML [2], and similarly WSMO can use WSML/XML [12]. Therefore, for transformation between the on-the-wire XML data and the semantic ontological data we can use an XML transformation language, for example XSLT [1].

With this approach, the grounding needs to link the inputs and outputs of the service with the appropriate XSLT transformations, and with these links in place, the actual run-time process is very simple: assuming an OWL-S client, for example, wants to send a message to a Web service, it serializes its OWL data in RDF/XML and then it runs the appropriate XSLT stylesheet, sending the result to the service. A message from the service goes through the inverse process: first it is transformed using a reverse XSLT stylesheet into RDF/XML, then the client parses the result directly into ontological data and processes it accordingly.

This approach is very simple and uses proven existing technologies, but it has a notable disadvantage: an XML representation of ontological data (like RDF/XML or WSML/XML) is often an unpredictable mixture of hierarchy and interlinking, as ontological data is not structured according to XML conventions (we say ontological data is not *native* XML data), so creating robust XSLT transformations for complex data structures may be a considerable task. With simple data, however, this problem is negligible, and since XSLT processors are readily available and many XML-savvy engineers have some XSLT experience, this approach is an ideal initial candidate for data grounding.

Of the two major SWS frameworks, OWL-S allows XSLT transformations for data grounding, whereas WSMO focuses on the more advanced approaches described below. WSDL-S also supports XML-level transformations.

3.2. Exchanging ontological data

The XSLT transformation approach described above makes use of the fact that SWS ontology languages can be serialized in XML. In certain situations this XML form could in fact be suitable for the WSDL interface of a Web service, for example for quick SWS prototyping, or when only semantic clients are expected to use a particular service, so the data will seldom be viewed as actual XML.

In these scenarios, the WSDL description of the messages will only indicate that they contain RDF/XML or WSML/XML, and the semantic description will say (as it does regardless of the grounding approach taken), what exact data goes in the inputs and outputs. The grounding only needs to link the semantic inputs and outputs with particular WSDL messages, which is done implicitly by all the behavior grounding approaches we describe in section 4.

This approach does not require any human designer to create grounding transformations, which may be a significant saving of effort. On the other hand, XML serializations of ontological data are not *native* XML data, so they may be hard to comprehend or hard to process by XML tools, and services that use this grounding approach may not integrate well with non-semantic Web services.

OWL-S allows this grounding style explicitly with the property `owlsParameter`, and WSMO grounding supports this style implicitly, when no transformation is specified between the ontological inputs and outputs and the WSDL messages of a service. WSDL-S also supports this style of data grounding.

3.3. XML Schema ad-hoc ontology

In this approach, an ad-hoc ontology is automatically generated from the schema of the XML messages, according to a conceptual mapping between the XML metamodel and the ontological metamodel. Along with this ad-hoc ontology, a set of mapping rules is also generated for the *lifting/lowering* transformations between the XML messages

and instances of the generated ontology.

In case the semantic description designer finds this generated ontology suitable for describing this particular Web service, this grounding can be fully automatic. On the other hand, if the generated ad-hoc ontology cannot be used, grounding involves an additional transformation between instances of the ad-hoc ontology and instances of the target ontology used by the service description. This transformation step uses ontology mediation approaches (e.g. [17]).

Similarly to the XSLT approach, the ad-hoc ontology approach has the benefit of reusing existing transformation technologies (ontology mediation in this case), and it also has the disadvantage that the generated ad-hoc ontology is not a *native* ontology (it is structured as a restrictive schema for data validation, as opposed to a descriptive ontology for knowledge representation), and this ontology can lack or even misrepresent semantics that are only implied in the XML. This can complicate the task of mediating between the ad-hoc ontology and the target ontology describing the service in a similar way as the *non-native* XML data can complicate the XSLT transformation.

Because WSMO puts heavy focus on mediation, WSMO grounding [14] is currently being developed in the direction of this approach, using WSMO ontology mediation for the transformations. WSDL-S can also support this approach, but OWL-S does not seem to support it.

3.4. Direct mapping language

Although we are not aware of any work in this direction in either of the SWS frameworks, we envision a third option that transforms between the XML data and the ontological data directly, using a specific transformation language.

While a new transformation language would have to be created, it could be optimized for the common transformation patterns between native ontological data and native XML, and so the manually created mappings could be simpler to understand, create and manage, than in the previous approaches. If the disadvantages of the other approaches prove substantial, this one should be considered.

4. Behavior grounding

In this paper, we define a *choreography model* of a Semantic Web Service framework as that part of the semantic description, which allows the client semantic processor to know what messages it can send or receive at any specific point during the interaction with a service. Choreography models also have other uses, for example detecting potential deadlocks, but these uses are outside the scope of this paper.

Because Semantic Web Services reuse WSDL, their choreography models must be tied to its simple model of separate operations, each one representing a limited message exchange. The ordering of messages within any single operation is defined by the operation's message exchange pattern, so the choreography model must specify in what sequence the operations can be invoked.

Choreography can be described explicitly, using some language that directly specifies the allowed order of operations. Conversely, choreography can also be described indirectly, by specifying the conditions under which operations can be invoked, and the effects of such invocations, and sometimes also the inputs and outputs of the service. Inputs, outputs, preconditions and effects are together commonly known as IOPEs.

OWL-S and WSDL-S both allow IOPEs to be specified on the level of WSDL operations. WSMO specifies the conditions and effects using abstract state machines. These two techniques have very different characteristics so we consider them in separate subsections.

First, section 4.1 details the OWL-S and WSDL-S approach to implicit choreography, then section 4.2 describes the WSMO approach with abstract state machines, and finally in section 4.3 we discuss the explicit choreography approach.

4.1. Planning with IOPEs

As we already mentioned, OWL-S and WSDL-S allow IOPEs (inputs, outputs, preconditions and effects) to be specified on the level of WSDL operations. With this information, known AI planning techniques [15] can be used to find a suitable ordering of the operations, based on the initial conditions and the end goal. In other words, the semantic client processor gets the description of IOPEs for all the available operations and then it plans the actual sequence it will use.

OWL-S does not ascribe IOPEs directly to WSDL operations, instead these conditions are attached to so-called atomic processes and OWL-S grounding binds these atomic processes to WSDL operations. This is all the grounding information that needs to be present with this approach, which makes the grounding very simple.

The main benefit of this implicit choreography approach is its flexibility, as different operation sequences can be chosen depending on the goal of a particular client, and the sequence can even be replanned in the middle of a run if some conditions unexpectedly change. However, planning algorithms usually have high computational complexity and require substantial resources, especially if there is a large number of available operations. In situations where the cost of AI planning is a problem, choreographies can be precomputed (or designed) for the supported goals, and these choreographies can then be described explicitly, as covered in the section 4.3.

4.2. WSMO choreography

In WSMO, choreography is modeled as an abstract state machine (ASM, cf. [11]), where the states are described with an ontology, and the client is (abstractly) allowed to read and write instances of certain concepts (classes in OWL terms), that are marked as "in", "out" or "shared" (both in and out). The state transition rules model the reactions to incoming data and the creation of outgoing data.

With this choreography model, WSMO is not limited to the message exchange view of Web services (cf. Triple Spaces [13] for an alternative), but grounding to WSDL becomes much more complex. To repeat, the purpose of grounding is to give the client all the necessary networking details so that it can successfully communicate with the service. When a client follows the WSMO choreography state machine, it knows from the activated transition rules that some particular data can be sent to the service. The grounding must now bind this data to a WSDL operation request, and if the operation has a response message, the client knows that it can expect specific data back.

When creating the grounding, the designer takes into account all the "in", "out" and "shared" concepts, as they are manipulated by the transition rules, and correlates "in" and "shared" concepts with input messages, and "out" and "shared" concepts with output messages. It may not be trivial to make sure that the expectations implied by the transition rules (some data will be created in response to some incoming data) match the operation message exchange patterns in the WSDL.

While the use of abstract state machines in this approach brings in the extensive results of ASM research (e.g. validation and verification), the intention mismatch between the WSDL model of operations and the ASM model of state transitions may complicate the task of a grounding designer. This approach should only be used if the benefits of ASM outweigh the added complexity.

4.3. Explicit choreography

An explicit choreography description uses some kind of process modeling language to specify the sequences of operations that are allowed on a particular Web service. The client processor must be able to discover the choreography description and then it simply follows what is prescribed.

OWL-S can describe choreographies explicitly with so-called *composite processes*, i.e. compositions of atomic processes. The composition ontology is based on various works in workflow and process modeling, and it contains constructs for describing the well-known patterns like sequence, condition and iteration. A client processor following an OWL-S composite process will simply execute the composition constructs, and grounding information will only be needed on the level of atomic processes, discussed in section 4.1. No other grounding information is necessary.

Alternatively to a SWS-specific composition language, a Web service choreography can be described with industrial languages like WSCI [9] or WS-CDL [4]. In this case it would be the goal of grounding to simply point from a Semantic Web Service description to the appropriate choreography document in WS-CDL or any other suitable language.

WSMO does not currently support any explicit choreography description, but if needed, an industrial choreography language can easily be adopted, as the grounding requirement of this approach is minimal — the pointer to a WS-CDL document, for example, can be implemented as a non-functional property of a service description in WSMO.

5. Conclusions and future work

Semantic Web Services frameworks are based on ontological models of Web services. These formal, logics-based models describe those aspects of Web services that are necessary for automating tasks like service discovery, negotiation, composition and invocation. On the other hand, the IT industry deploys Web services with only limited description capabilities, sufficient when Web services are mostly used, composed and deployed manually. The industry standard for Web service description is WSDL, a language that views Web service interfaces as collections of operations, combined with XML Schema, a validation-oriented language for describing XML data structures. The meaning and allowed ordering of the operations, and the meaning of the data is usually specified in plain text, if at all, exposing only the intuitive meanings of the names of the XML elements and WSDL operations and interfaces.

The SWS research community recognizes the importance of interoperability with the industry Web services, so the ontological descriptions of Web services are *grounded* in WSDL and XML Schema. Semantic Web Services can thus be used manually with existing Web service toolkits, and existing syntactic Web services can be described semantically and used automatically in SWS environments.

In this paper we discussed the bridge between the syntactic and the semantic descriptions — the grounding. There are two areas where grounding is necessary: data mapping between ontologies and XML, and providing networking details for service invocation. For both areas we have identified several grounding approaches along with their advantages and disadvantages.

Table 1 summarizes how the grounding approaches are supported in the two SWS frameworks we have considered (OWL-S and WSMO). WSDL-S is also contained in the table, even though it is not a full SWS framework. WSDL-S is very broad but does not itself contain any concrete semantic tools, therefore all the capabilities in the WSDL-S

Table 1. Groundings in SWS frameworks

Approach	OWL-S	WSMO	WSDL-S
Data grounding			
XML-level	+	-	*
Ontological data	+	+	*
Generated ontology	-	+	*
Direct mapping	-	-	*
Behavior grounding			
Planning	+	-	*
ASMs	-	+	-
Explicit chor.	+	-	-

column exist only *in potentia*.

We have not resolved the question of where grounding information should be specified, i.e. as extensions to WSDL or as specific links in the semantic description documents, but both approaches can be useful. While currently both major SWS frameworks put grounding in the semantic description, WSDL-S presents several WSDL hooks for semantic information, and it can potentially be used with both mentioned SWS frameworks. Work on integrating WSDL-S with WSMO is underway; we are not aware of similar work towards integrating WSDL-S with OWL-S.

Data grounding is the main part of SWS grounding, as there are multiple different approaches to one complex problem — how to transform efficiently between XML and ontological data. The current SWS frameworks take different approaches, and we propose a new one with direct mapping language, which may result in the most natural (and therefore the most manageable) mapping descriptions, even though it does involve creating a new data transformation language, which is a considerable task.

Behavior grounding has different grounding approaches, too, but they vary because of the differences in the underlying choreography models. The OWL-S grounding approach simply links OWL-S atomic processes with WSDL operations; the proposed grounding to choreography described in WS-CDL or WSCI simply links the semantic description of a Web service to the choreography document. Only the WSMO grounding is more complex, mainly due to the mismatch between the choreography model in WSMO and the interface and operations model in WSDL. This complexity is, however, the price to be paid for the flexibility of WSMO choreography model which is not limited to the usual messaging view of Web services. Recognizing that WSMO choreography is work in progress, we suggest that WSMO should add a choreography model that will allow simpler grounding, as the overall complexity in WSMO choreography may hamper adoption.

The authors are involved in WSMO grounding effort, and will focus their future work on extending WSMO with the various approaches, especially for data grounding.

References

[1] XSL Transformations. Recommendation, W3C, November 1999. Available at http://www.w3.org/TR/xslt.

[2] RDF/XML Syntax Specification (Revised). Recommendation, W3C, February 2004. Available at http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

[3] UDDI Version 3.0.2. OASIS Standard, Oct 2004. Available at http://www.oasis-open.org/committees/uddi-spec/doc/spec/v3/uddi-v3.0.2-20041019.htm.

[4] Web Services Choreography Description Language Version 1.0. Working Draft, W3C, December 2004. Available at http://www.w3.org/TR/2004/WD-ws-cdl-10-20041217/.

[5] XML Schema Part 1: Structures. Recommendation, W3C, October 2004. Available at http://www.w3.org/TR/xmlschema-1/.

[6] Web Services Description Language (WSDL) Version 2.0. Last Call Working Draft, W3C WS Description Working Group, August 2005. Available at http://www.w3.org/TR/2005/WD-wsdl20-20050803.

[7] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M. Schmidt, A. Sheth, and K. Verma. Web Service Semantics - WSDL-S. Technical note, April 2005. Available at http://lsdis.cs.uga.edu/library/download/WSDL-S-V1.html.

[8] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services*. Springer, 2003.

[9] A. Arkin, S. Askary, S. Fordin, W. Jekeli, K. Kawaguchi, D. Orchard, S. Pogliani, K. Riemer, S. Struble, P. Takacsi-Nagy, I. Trickovic, and S. Zimek. Web Services Choreography Interface 1.0, June 2002. Available at http://www.w3.org/ TR/wsci/.

[10] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[11] E. Borger and R. F. Stark. *Abstract State Machines: A Method for High-Level System Design and Analysis*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[12] J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel. The Web Service Modeling Language WSML. Available at http://www.wsmo.org/TR/d16/d16.1/v0.3/20051005/, October 2005.

[13] D. Fensel. Triple-space computing: Semantic Web Services based on persistent publication of information. In *Proceedings of the IFIP Int'l Conference on Intelligence in Communication Systems, Bangkok, Thailand*, pages 43–53, 2004.

[14] J. Kopecký, M. Moran, D. Roman, and A. Mocan. WSMO Grounding. Available at http://www.wsmo.org/TR/d24/d24.2/v0.1/20050916/, September 2005.

[15] D. Nau, M. Ghallab, and P. Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[16] D. Roman, U. Keller, H. Lausen, J. de Brujin, R. Lara, M. Stollberg, A. Polleres, C. Bussler, and D. Fensel. Web service modeling ontology. *Applied Ontology*, 1(1), 2005. To appear.

[17] F. Scharffe and J. de Bruijn. A Language to Specify Mappings Between Ontologies. In *Proc. of the Internet Based Systems IEEE Conference (SITIS05)*, 2005.

[18] The OWL Services Coalition. OWL-S 1.1 release. Available at http://www.daml.org/services/owl-s/1.1/, November 2004.