Bringing Semantic Annotations to Web Services: OWL-S from the SAWSDL Perspective

David Martin[1], Massimo Paolucci[2], Matthias Wagner[2]
[1] *Artificial Intelligence Center, SRI International*
martin@ai.sri.com
[2] *DoCoMo Communications Laboratories Europe GmbH*
{paolucci,wagner}@docomolab-euro.com

Abstract

Recently the World Wide Web Consortium (W3C) produced a standard set of "Semantic Annotations for WSDL and XML Schema" (SAWSDL). SAWSDL provides a standard means by which WSDL documents can be related to semantic descriptions, such as those provided by OWL-S (OWL for Services) and other Semantic Web services frameworks. We argue that the value of SAWSDL cannot be realized until its use is specified, and its benefits explained, in connection with a particular framework. This paper is an important first step toward meeting that need, with respect to OWL-S. We explain what OWL-S constructs are appropriate for use with the various SAWSDL annotations, and provide a rationale and guidelines for their use. In addition, we discuss some weaknesses of SAWSDL, and identify some ways in which OWL-S could evolve so as to integrate more smoothly with SAWSDL.

1. INTRODUCTION

The driving objective behind Web services technologies, such as the Web Services Description Language (WSDL) [2], is to provide reliable, ubiquitous software interoperability across platforms, across networks, and across organizations. Accordingly, the primary technical focus has been on standardizing and validating the syntax and mechanisms of message exchange, so as to support reliable, vendor-neutral communications between Web services and their users.

Semantic Web services technology aims to provide for richer semantic specifications of Web services, so as to enable fuller, more flexible automation of service provision and use, to support the construction of more powerful tools and methodologies, and to promote the use of semantically well-founded reasoning about services. The field, which got under way around 2001 [14], includes substantial bodies of work, such as the efforts around OWL for Services (OWL-S) [11], the Web Services Modeling Ontology (WSMO) [10], and METEOR-S [18].

Each of these efforts has sought to build out from, or integrate with, WSDL, rather

than reinventing that part of the Web services picture. This has resulted in several distinct, ad hoc, styles of integration with WSDL. Recently, however, the World Wide Web Consortium (W3C) produced a standard set of "Semantic Annotations for WSDL and XML Schema" (SAWSDL) [6]. SAWSDL, based primarily on the earlier work on WSDL-S [1], provides a standard means by which WSDL documents can be related to semantic descriptions, such as those provided by OWL-S and WSMO.

SAWSDL represents a conservative, incremental approach to introducing semantic characterization of Web services into mainstream Web service practices. Its objectives are modest. For example, it aims to provide semantic characterization of a service's input and output types, which can be useful in disambiguating those types in the context of simple forms of service discovery. But it does not attempt to provide a comprehensive framework to support more sophisticated approaches to discovery, composition, or any of the other service-related tasks that Semantic Web services research aims to automate.

SAWSDL does not specify a particular semantic framework within which to characterize the semantics of Web services. Rather, it defines a small set of WSDL extension attributes, which may be used to refer to constructs within *any* external semantic framework. SAWSDL is completely noncommittal regarding the choice of semantic framework. It is important to understand, however, that SAWSDL is of very little use unless there is an additional specification of conventions and guidelines for what can be referred to in a particular semantic framework, and what it means to do so. Consequently, such a specification is an essential and timely next step in bringing Semantic Web services research to fruition.

In addition to discovery, the SAWSDL specification mentions that SAWSDL annotations can be used during composition and invocation (Sections 1 and 2 of [6]). However, the specification says essentially nothing about how these tasks are to be supported or what degree of automation may be achieved. Indeed, there is very little that can be said, without reference to a particular semantic framework. (As we will see, the intended use of the schema mapping attributes *is* pretty clear, because they are more specialized than the modelReference attribute. However, even in this case, no guidance can be given regarding the details of how to specify a mapping, without reference to a particular semantic framework.)

In this paper, we provide guidelines regarding the use of OWL-S in conjunction with SAWSDL. These guidelines are provided *from the SAWSDL perspective.* That is, we do not try to explain everything that can be done with OWL-S in conjunction with WSDL. Rather, we simply explain what OWL-S constructs are appropriate for use with the various SAWSDL annotations. These explanations are provided with a view to supporting WSDL users and WSDL tool vendors in achieving the kinds of objectives that are associated with SAWSDL.

An analysis with similar objectives has previously been given for using WSMO with WSDL-S [8], and the use of SAWSDL is beginning to appear in WSMO tools such as [5].

Because of space limitations, it is not possible to give an adequate overview of WSDL or OWL-S. For introductory material on WSDL, the reader is referred to [2]. In Section 2,

we give a brief characterization of OWL-S. Section 3 discusses the use of SAWSDL's modelReference attribute with OWL-S, and Section 4 discusses the use of SAWSDL's schema mapping attributes. In Section 5, we discuss some overarching issues and summarize our recommendations. Section 6 concludes.

2. OWL-S

As noted in [12], the principal high-level objectives of OWL-S are (i) to provide a general-purpose representational framework in which to describe Web Services; (ii) to support automation of service management and use by software agents; (iii) to build, in an integral fashion, on existing Web Service standards and existing Semantic Web standards; and (iv) to be comprehensive enough to support the entire life cycle of service tasks.

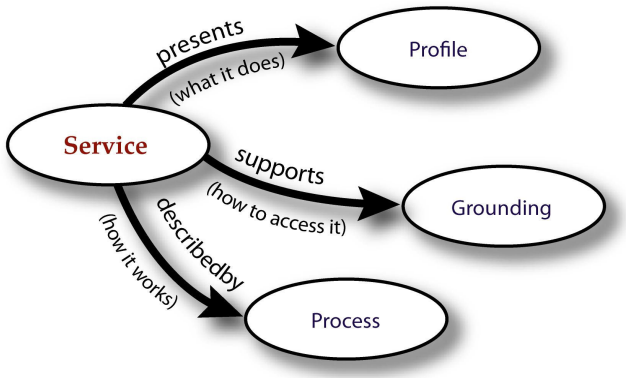

Figure 1. ***Top level of the OWL-S service ontology.***

OWL-S (formerly known as DAML-S) is an OWL ontology [13] that includes three primary subontologies: the service profile, process model, and grounding. The service profile is used to describe *what the service does*; the process model is used to describe *how the service is used*; and the grounding is used to describe *how to interact with the service*. The service profile and process model are thought of as *abstract* characterizations of a service, whereas the grounding makes it possible to interact with a service by providing the necessary *concrete* details related to message format, transport protocol, and so on. Figure 1 shows the relationships between the top-level classes of the ontology. In this figure, an oval represents an OWL class, and an arc represents an OWL property. For example, the *presents* property represents a relationship that can hold between a *Service* and a *Profile.*[1]

[1] For ease of exposition, Figure 1 presents a slight simplification. In particular, it omits an organizational layer of classes named `ServiceProfile`, `ServiceGrounding`, and `ServiceModel`.

Each service described using OWL-S is represented by an instance of the OWL class *Service,* which has properties that associate it with a process model (an instance of the class *Process)*, one or more groundings (each an instance of the class *Grounding)*, and optionally one or more profiles (each an instance of the class *Profile*). A process model provides the complete, canonical description of how to interact with the service at an abstract level, and the grounding supplies the details of how to embody those interactions in real messages to and from the service. Each service profile may be thought of as a summary of salient aspects of the process model plus additional information that is suitable for the purposes of advertising and selection. Several different types of grounding have been devised for OWL-S. The default and most widely used grounding, which is included in the OWL-S releases, employs WSDL. [11] discusses the grounding to WSDL 1.1, and [16] presents a proposal for a grounding that employs WSDL 2.0 and SAWSDL.

In this paper, we are concerned with the use of constructs of the profile and process model as referents of SAWSDL annotations. Because this paper adopts a perspective centered around WSDL and SAWSDL, there is no need to employ the OWL-S grounding as a source of referents. OWL-S's grounding reflects an OWL-S perspective; that is, it is motivated by use cases in which service processing, tools, and reasoning of various kinds are organized around OWL-S. For example, the OWL-S Virtual Machine [15] executes OWL-S process models. When an invocation of an external Web service is indicated in a process model, the Virtual Machine uses the grounding to arrange for the invocation of that Web service.

Here, by contrast, we do not assume that processing will be organized around OWL-S. While the guidelines given here are consistent with the OWL-S grounding, they are meant to support the use of semantics in a manner that builds incrementally on WSDL usage, tools, and environments, in keeping with the philosophy underlying SAWSDL. At the same time, we strive to be as general as possible, and to support a variety of service-related tasks in a variety of architectures.

3. USING THE *modelReference* ANNOTATION

SAWSDL introduces three new extension attributes for use in WSDL and XML Schema documents, and discusses some of their possible uses [6]. modelReference can be used in both WSDL and XML Schema documents. The schema mapping attributes, liftingSchemaMapping and loweringSchemaMapping, are intended for use only in XML Schema documents. The addition of these attributes requires no other changes to existing WSDL or XML Schema documents, or the manner in which they had been used previously. In this section we discuss how modelReference can be used with OWL-S.

The SAWSDL specification states that "A model reference may be used with every element within WSDL. However, SAWSDL defines its meaning only for wsdl:interface,

wsdl:operation, wsdl:fault, xs:element, xs:complexType, xs:simpleType and xs:attribute"[2] [6]. Here, we discuss the OWL-S constructs that are appropriate as referents of modelReference in each of these settings. We begin with operations, interfaces, and faults. Then we turn to the XML Schema elements (those with the "xs" prefix). Following that we discuss possible uses of modelReference with input and output (message) elements (even though those uses are not defined by SAWSDL).

3.1 Operations

In WSDL, an operation represents "a simple interaction between the client and the service. Each operation specifies the types of messages that the service can send or receive as part of that operation. Each operation also specifies a message exchange pattern [MEP] that indicates the sequence in which the associated messages are to be transmitted between the parties" [2].

Conceptually, the atomic process of OWL-S corresponds very closely to WSDL's operation, and this correspondence was one of the cornerstones of OWL-S's grounding to WSDL 1.1 [11]. For example, an operation that takes a single input message, and outputs a single output message, exhibits the same behavior as an OWL-S atomic process with a single input and a single output. In many cases such as this it is straightforward to establish a mapping between the constituents of the operation and those of the atomic process. In these straightforward cases, the value of modelReference should be the URI of an atomic process. Then, as described in Sections 3.4 and 3.5, the payload of the messages can be mapped to the types of the atomic process's inputs and outputs (or to the inputs and outputs themselves), using modelReference annotations of the relevant XML Schema (or WSDL) declarations.

However, there is a very important caveat regarding the mapping of an operation to an atomic process: it can work only for simple message exchange patterns. The atomic process is defined in terms of a (possibly empty) set of inputs (arriving simultaneously) followed by a (possibly empty) set of outputs (leaving simultaneously). If an MEP cannot be mapped into that simple sequence of events, then that MEP cannot be mapped onto the I/O of an atomic process. (One could imagine a partial mapping, where some messages were ignored, but we will not consider that possibility here.)

WSDL 2.0 provides eight predefined MEPs: In-Only, Robust In-Only, In-Out, In-Optional-Out, Out-Only, Robust Out-Only, Out-In, and Out-Optional-In [3]. Four of these MEPs – In-Only, In-Out, Out-Only, Robust Out-Only – can be mapped onto the I/O of an atomic process.

---

[2] It should be noted that the guidance given regarding the uses of modelReference for each of these elements has much more the flavor of *suggestions* than *definitions*. For example, the material on usage with interfaces mentions that modelReference can be used "to categorize them according to some model, specify behavioral aspects *or other semantic definitions* [emphasis added]", and similarly for operations.

What about Out-In, Out-Optional-In, Robust-In-Only, In-Optional-Out, and other, more complex MEPs that cannot be mapped onto the I/O of an atomic process? In principle, they can be mapped onto composite processes. Indeed, for any MEP, it is possible to construct an OWL-S composite process that supports (and requires) the same pattern of inputs and outputs. Similarly, for any OWL-S composite process, it is possible to construct a message pattern that corresponds to the I/O behavior of that composite process.

However, there are some issues needing further attention, having to do with the mapping of the inputs and outputs, which we take up in Section 3.5.

Finally, let us also note that it may be useful in some situations to annotate an operation by referring to a *profile*. If one is primarily concerned with *categorizing* operations as to the functionality they provide, an OWL-S profile is more appropriate for that purpose than a process. Nevertheless, the process should be regarded as the most natural referent for an operation, for all the reasons given above. In the following subsection, we discuss another possible use of the OWL-S profile as a referent.

3.2 Interfaces

In WSDL, an interface is, in essence, a group of related operations. The WSDL specification is not specific about how these operations are related, except to say that they make up the abstract interface of a Web service: "A WSDL 2.0 interface defines the abstract interface of a Web service as a set of abstract operations, each operation representing a simple interaction between the client and the service." A service, in turn, "specifies a single interface that the service will support, and a list of endpoint locations where that service can be accessed."

It should be noted that there is a mismatch between WSDL's notion of "service" and that of OWL-S. In OWL-S, the Service class is an organizational unit that packages up the information that describes a single process; that process is, in effect, the essence of the service. As noted above, OWL-S's Process corresponds to WSDL's operation. Hence, an OWL-S Service also corresponds best to WSDL's operation, rather than WSDL's service.

Indeed, OWL-S does not have a construct for grouping processes. Therefore, it does not at present have a construct that corresponds *directly* (structurally) to WSDL's interface or WSDL's service. This is an area under consideration for a future release of OWL-S.

Nevertheless, there are three possible ways in which an interface's modelReference can meaningfully refer to an OWL-S construct (or constructs). The first of these is to be preferred, given the intent that is expressed in the SAWSDL specification for these annotations.

(1) The SAWSDL specification indicates a possible use of the interface modelReference for *categorization* purposes. It mentions, as an example, an interface annotation that refers to an "electronics" concept in some semantic model. (This example provides an extremely limited bit of information – that is, that the interface has something

to do with electronics. No doubt one could do better, for example, by referencing a concept for "ElectronicsRepairService" or "ElectronicsForSale".)

In fact, the OWL-S Profile is meant to be used for categorization. To do this, one takes advantage, in a very natural way, of OWL's mechanisms for building a class hierarchy; that is, a hierarchy of subclasses of Profile. For example, one might have RailTicketSales as a subclass of TravelTicketSales, as a subclass of TravelAgency, which in turn is a subclass of Profile. To represent a specific rail ticketing service, one would create an instance (i.e., OWL individual) of the class RailTicketSales. A larger, more comprehensive class hierarchy of this kind can be used as the basis for a "yellow pages" registry of services. An instance of a profile class from such a hierarchy can serve as the referent of the modelReference annotation of an interface. If a particular instance is not available, the class itself can serve as the referent.

It should be noted that an instance of OWL-S profile normally is bundled with a process model and a grounding, but that is not required by OWL-S.

(2) Since modelReference always allows for a *list* of URIs, one can simply list all the URIs of the processes that correspond to the interface's operations. This information, however, would be redundant with the modelReference annotations of the operations themselves, so that limits the value of this approach.

(3) In some cases it is reasonable to map an interface to a composite process. A composite process can be viewed as a grouping mechanism, because it specifies and coordinates calls (in the form of Perform statements) to a number of atomic processes. In certain cases, it could make sense to regard this set of atomic processes that are called by a composite process as the correlate of a WSDL interface. However, this cannot be regarded as a general rule, because in general the relationship between the atomic processes called from a composite process is quite different from the relationship between the operations grouped into an interface.

3.3 Faults

OWL-S does not yet have a concept of *fault* (or *exception*) *per se.* However, OWL-S has the *conditional effect,* which can be used to capture the same intent. A conditional effect (of a process) simply states what effects will occur under a given condition. That condition can directly correspond to a fault, such as the "ItemUnavailable" fault example given in Section 3.3 of [6]. Thus, it makes sense for the modelReference of a fault to refer to a conditional effect.[3]

3.4 XML Schema Elements

By default, and in what is by far the most common usage, the content of a WSDL message

[3] Precisely speaking, a conditional effect is an instance of OWL-S's *Result* class.

is described using XML Schema. That is, XML Schema is used to define an *element*, which in turn is associated with a message of a WSDL operation. The element defines the syntax that is allowed for the content of the associated message. The XML Schema definitions can appear inline, in the *types* section of a WSDL document, or in a separate XML Schema document that gets imported by the WSDL document.

The SAWSDL charter [9] gives a motivational example in which an operation, having input and output messages "*amount* and *tax*, both of type *xs:double*, could have different meanings: calculation of tax on a product, calculation of income tax, etc." The problem illustrated by this example, of course, is that a very general, ubiquitous I/O type like *xs:double* tells you very little about the functionality or usage associated with an operation using that type. Here, an annotation referring to a type (e.g., a *SalesTax* concept) defined in some semantic framework, such as OWL, can provide value by helping to discover operations that can meet a given set of requirements.

To support this kind of use case, SAWSDL allows for the annotation of any *xs:element, xs:complexType, xs:simpleType,* or *xs:attribute* definition. For our purposes, there is little difference between these four kinds of definitions[4]; in each case, a modelReference annotation will associate a semantically defined concept with the corresponding unit of structure in XML Schema. In general, it is straightforward to map from a unit of structure in XML Schema to an OWL concept – and there can be a good deal of flexibility in doing so. In many cases, an element (or complex or simple type) such as, for example, PurchaseOrder, will map naturally onto an OWL class with similar structure. In other cases, depending on the choices that have been made in structuring the ontology, it could also be reasonable to map an element (or complex or simple type) onto an OWL individual. In the case of a complex type, the SAWSDL specification notes that it can be annotated in a top-down style, a bottom-up style, or a combination of the two. Thus, in many cases, a complex type could map very naturally onto an OWL class, and its nested types could map onto the types (ranges) of that class's properties, that is, assuming that the XML Schema type and the OWL class have a parallel structure. But SAWSDL does not assume a parallel structure; indeed, SAWSDL is explicitly noncommittal regarding the relationship between the high-level and the lower-level annotations within a complex type: "A complex type can be annotated at both the top and member level. These annotations are independent of each other" (Section 4.1.2 of [6]).

3.5 Input and Output Elements

As described above, SAWSDL defines the use of modelReference with several kinds of XML Schema declarations. This gives an effective means of mapping from XML Schema to OWL. That is, given an arbitrary unit of structure defined in XML Schema, SAWSDL allows you to associate it with any OWL entity (or with a list of OWL entities) that can be

[4] It should be noted that, "in WSDL 2.0, all normal and fault message types must be defined as single elements at the topmost level (though of course each element may have any amount of substructure inside it)" [2].

referenced by URI. (SAWSDL has nothing to say about mapping in the other direction, and that is also out of scope for this paper.)

However, it is important to recognize the limitations of this approach in the context of *services*. The inputs and outputs of services are carried in messages. In WSDL, messages are described using MEPs, and *input* and *output* elements associated with operations. But SAWSDL's XML Schema annotations deal only with *content,* and say nothing about inputs, outputs, or message exchange patterns (MEPs).

Why does this matter? After all, it is certainly true that, with SAWSDL's defined XML Schema annotations, the content of any input or output message (and any element of structure within that content) can be mapped to a semantic referent. The problem is simply that the XML Schema annotations are not adequate to provide full disambiguation of the semantics associated with inputs and outputs – at least not without forcing a cumbersome duplication of XML Schema declarations.

Consider, for example, company X's use of a WSDL document (developed before SAWSDL was available) that defines an XML element *PurchaseOrder,* and reuses that element as the input type of three different operations. Suppose one of those operations uses the *PurchaseOrder* element to carry information for a *new* purchase, whereas another operation uses that same *PurchaseOrder* element to carry information for a *modified* purchase, and yet another operation uses it for a purchase to be *cancelled.* Suppose, further, that company X develops an OWL ontology that has distinct classes for *NewPurchaseOrder, ModifiedPurchaseOrder,* and *CancelledPurchaseOrder*, and wants to use its ontology to annotate its existing services. In a scenario such as this, the service could not be properly annotated without defining *the same three distinctions* in XML Schema, as distinct elements. Having done this, modelReference could be used, in the element declarations, to refer to the three different OWL classes appropriately. To properly correlate the three new elements, and their annotations, with the operations, the input constructs within the operation definitions would also have to be modified to indicate which of the elements is used with which operation. This is a workable solution, but at the cost of considerable effort in maintaining legacy services. Given all this required effort at capturing these distinctions in XML Schema, one might well wonder if the semantic annotations are adding any value.

Moreover, when complex MEPs are used, difficulties such as these can arise within the annotation of a *single* operation. If an MEP has more than one input message, a similar situation could arise, in which multiple input messages could carry content of the same XML Schema type, but it would be important to annotate them with different semantic referents.[5] SAWSDL's defined uses of modelReference (with operation, interface, fault, and XML Schema constructs) do not readily allow for this. For many purposes, this can be a serious limitation. Even discovery is not well supported anymore. Consider an operation A with an MEP that takes an input message with a semantic referent of X, followed by an input message, using the same XML Schema type, with a semantic referent of Y.

[5] Similar difficulties can arise with outputs, of course.

Operation B also takes two input messages using the same XML Schema type, but with semantic referents of Y followed by X. If these two operations cannot be distinguished, discovery becomes much less effective.

This situation can be remedied by setting out some guidelines for the use of modelReference with WSDL's *input* and *output* (message) constructs. With the use of these constructs, the purchase order example above can be easily accommodated by adding modelReferences (pointing to *NewPurchaseOrder, ModifiedPurchaseOrder,* and *CancelledPurchaseOrder)* directly onto the WSDL input declarations of the three operations, as appropriate. Examples with complex MEPs can similarly be disambiguated.

Instead of using an OWL *class* as the referent of an input (or output) element's modelReference, it is also possible to use an OWL-S input (or output) construct, or a set of OWL-S input (or output) constructs. These OWL-S constructs should, of course, belong to the process that corresponds to the operation of the message. Indeed, this is a more natural mapping for WSDL input and output elements. Compared to the use of OWL classes as referents, there is no lost information, because each OWL-S input and output already includes a mention of the class that serves as the type of the input or output.

4. USING SCHEMA MAPPING ANNOTATIONS

SAWSDL's schema mapping annotations, *liftingSchemaMapping* and *loweringSchemaMapping*, "are used to associate a schema type or element with a mapping to an ontology …. The value of the *liftingSchemaMapping* attribute is a set of zero or more URIs that reference mapping definitions. A mapping referenced by this attribute defines how an XML instance document conforming to the element or type defined in a schema is transformed to data that conforms to some semantic model" (Section 4.2 of [6]). Similarly, *loweringSchemaMapping* is used to reference a mapping from data expressed in a semantic model to data expressed in an XML document.

There is very little to say about the schema mapping annotations that is specific to OWL or OWL-S. These annotations are likely to be used in conjunction with XSLT primarily, but the SAWSDL specification does not require XSLT or any other particular mapping language. The schema mapping annotations are the only aspects of SAWSDL that are clearly intended for use at runtime (and only at runtime). It should be noted that the OWL-S 1.1 (and previous release) groundings have also made use of XSLT scripts in the same general manner. As explained in [16], the use of an XSLT (or similar) syntax-based transformation approach from OWL to XML is problematic, because there are generally a number of different ways that the same content can be serialized in OWL. It can be quite complicated to write an XSLT script that handles all the different variants.

The OWL-S 1.1 grounding adopted some measures to alleviate this problem. It allows for the use of precondition expressions to bind variables to values in the semantic model (typically values passed in as inputs), and it specifies that a runtime environment should

pass these bindings into corresponding variables declared in XSLT. The extent to which this alleviates the problem will depend upon the OWL-S developer—specifically, on the manner in which he or she writes precondition expressions. Preconditions can be written in a variety of languages, including SWRL [7] and SPARQL [17]. In principle, it is possible to use preconditions to break down complex OWL individuals into primitive elements, thus avoiding the issue of handling multiple possible serializations.

5. DISCUSSION

The SAWSDL specification leaves a great deal to the imagination, and it remains to be seen whether, and in what ways, it will come to be widely used. There is very little that can be done with SAWSDL that does not require additional conventions. This paper is meant to be a start toward a set of conventions for using SAWSDL with OWL-S as the source of annotation referents.

This need for additional conventions is perhaps most evident with respect to SAWSDL's schema mappings (lifting and lowering). More than any of SAWSDL's other annotations, it is quite clear that the schema mappings cannot stand alone. That is, additional specifications and machinery are needed for them to be useful. At a minimum, a tool will need to know where to get the semantic data that needs to be lowered, or where to deliver the result of a lifting operation.[6] Of course, a great many other details may need to be specified as well, depending on purpose and context. For example, as noted earlier, OWL-S relies on variable bindings to be propagated from preconditions (normally expressed in SPARQL) to XSLT. Propagating these bindings cannot be accommodated, much less specified, using a schema mapping annotation – because these annotations allow for nothing other than a reference to a mapping script, such as an XSLT script. The conventions for variable bindings from another framework cannot themselves be captured in XSLT; they require additional specification.

This is an illustration of the inherent weakness of SAWSDL with respect to more ambitious use cases. These more ambitious scenarios lead one to the conclusion that additional steps beyond SAWSDL will be needed before long; that is, conventions for use with larger frameworks. This conclusion is the motivation for the full OWL-S grounding, discussed in [16], which builds on SAWSDL annotations.

It is important to note that implicit constraints will often be associated with the use of SAWSDL annotations, if they are to be used in a coherent fashion with a single semantic framework such as OWL-S. For example, SAWSDL annotations of XML Schema elements can be used independently of services; they can be used merely to establish correspondences between elements of XML Schema definitions and elements of OWL ontologies. However, in the context of a larger, semantically annotated, WSDL document,

[6] With model references, at least, one can imagine getting some mileage simply by comparing their URIs.

an implicit set of constraints is associated with these annotations of XML Schema. This is because of the way in which the XML Schema types are used with operations, on the WSDL side, and the corresponding OWL types are used with the inputs and outputs of atomic processes, on the OWL-S side. Once you have mapped an operation to an atomic process, you have also implicitly established a correspondence between the set of XML Schema types used as the I/O types of the operation, and the set of OWL classes used as the I/O types of the atomic processes. To maintain coherence, then, these types need to be used consistently on both sides, across all operations and processes. Constraints such as these could and should be checked by tools.

OWL-S needs to evolve to support faults in a more straightforward manner. It would also be helpful if OWL-S had an organizational construct that directly correlated to WSDL's notion of an interface as a collection of operations.

6. CONCLUSION

We have given a rationale and guidelines for the use of OWL-S constructs as the referents of SAWSDL annotations. As explained in the Introduction, this kind of coupling of SAWSDL with a particular semantic framework is an essential and timely next step in bringing Semantic Web services research to fruition. Here is a summary of our recommendations:

- The modelReference of a WSDL operation can refer to an OWL-S atomic or composite process. With simple MEPs, either an atomic process or a composite process can be used (assuming that the process supports a pattern of I/O that is equivalent to the MEP). With complex MEPs (as characterized in Section 3.1), only a composite process can be used.
- The modelReference of a WSDL interface should refer to an instance of an OWL-S profile class (i.e., Profile or a subclass of Profile). If a particular instance is not available, a profile class can serve as the referent.
- The modelReference of a WSDL fault should refer to a conditional effect of an OWL-S process – the process that corresponds to the operation for which the fault is declared.
- Model references in XML Schema should refer to OWL constructs, and can do so independently of OWL-S.
- In addition, model references on WSDL input and output elements should be used to relate those elements to inputs and outputs of an OWL-S process – the process that corresponds to the operation for which the input or output element is declared.
- Schema mapping (lifting and lowering) annotations can refer to XSLT scripts. However, the usefulness of these scripts in translating from OWL is limited, as discussed in Section 4.

This paper and these recommendations assume that the most complete possible mapping is desired from WSDL onto OWL-S. A complete annotation of a WSDL document implies a number of constraints on the relationships between the referents of the annotations. Some of these constraints, which are not made explicit in SAWSDL, have been discussed here. They can and should be checked by tools.

These recommendations are, intentionally, of maximum generality so as to be relevant to a wide variety of use cases, purposes, and environments, and are provided here with a view to supporting WSDL users and tool vendors who want to use SAWSDL as the basis for an incremental introduction of semantics into Web service usage. More detailed specifications can evolve from these recommendations to support different situations, tool designs, etc.

Acknowledgements

Many thanks to Jacek Kopecky for his knowledgeable and insightful comments.

References

[1] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M. T. Schmidt, A. Sheth and K. Verma, “Web Service Semantics - WSDL-S”, W3C Member Submission, 7 November 2005; http://www.w3.org/Submission/WSDL-S/.

[2] D. Booth and K. Canyang, “Web Services Description Language (WSDL) Version 2.0 Part 0: Primer”, W3C Candidate Recommendation 27 March 2006; http://www.w3.org/TR/2006/CR-wsdl20-primer-20060327/.

[3] R. Chinnici, H. Haas, A. Lewis, J. Moreau, D. Orchard, and S. Weerawarana, Eds., “Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts”, W3C Candidate Recommendation 27 March 2006; http://www.w3.org/TR/2006/CR-wsdl20-adjuncts-20060327/.

[4] R. Chinnici, J. Moreau, A. Ryman, and S. Weerawarana, Eds., “Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language”, W3C Candidate Recommendation 27 March 2006; http://www.w3.org/TR/2006/CR-wsdl20-20060327/.

[5] M. Dimitrov, A. Simov, M. Konstantinov, L. Cekov, and V. Momtchev, “WSMO Studio Users Guide”, May 11, 2007, available at http://www.wsmostudio.org/doc/wsmo-studio-ug.pdf.

[6] J. Farrell and H. Lausen, Eds., "Semantic Annotations for WSDL and XML Schema", W3C Candidate Recommendation 26 January 2007; http://www.w3.org/TR/2007/CR-sawsdl-20070126/.

[7] I. Horrocks, P. F. Patel-Schneider, H. Boley, et al., "SWRL: A Semantic Web Rule Language Combining OWL and RuleML", W3C Member Submission, 21 May 2004; http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.

[8] J. Kopecký, "Aligning WSMO and WSDL-S", WSMO Working Draft, 5 August 2005; http://www.wsmo.org/TR/d30/v0.1/.

[9] J. Kopecký, Chair, C. Bournez, E. Prud'hommeaux, Team Contacts, "Semantic Annotations for WSDL Working Group Charter", Oct. 2005; http://www.w3.org/2005/10/sa-ws-charter.

[10] H. Lausen, A. Polleres, and D. Roman, "Web Service Modeling Ontology (WSMO)", W3C Member Submission, 2005; http://www.w3.org/Submission/WSMO/.

[11] D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara, "OWL-S: Semantic Markup for Web Services", W3C Member Submission, Nov. 2004; http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/.

[12] D. Martin, M. Burstein, D. McDermott, S. McIlraith, M. Paolucci, K. Sycara, D. McGuinness, E. Sirin, and N. Srinivasan, "Bringing Semantics to Web Services with OWL-S", *World Wide Web Journal*, Vol. 10, No. 3, Sept. 2007.

[13] D. L. McGuinness and F. van Harmelen, "OWL Web Ontology Language Overview", W3C Recommendation, 10 February 2004; http://www.w3.org/TR/2004/REC-owl-features-20040210/.

[14] S. McIlraith, T.C. Son, and H. Zeng, "Semantic Web Services", *IEEE Intelligent Systems*, Vol. 16, No. 2, pp. 46–53, March-April 2001.

[15] M. Paolucci, A. Ankolekar, N. Srinivasan, et al., "The DAML-S Virtual Machine", in *Proceedings of the 2nd International Semantic Web Conference (ISWC)*, Sanibel Island, FL, USA, October 2003, pp. 335–350.

[16] M. Paolucci, M. Wagner, and D. Martin, "Grounding OWL-S in SAWSDL", in *Proceedings of the International Conference on Service Oriented Computing (ICSOC 2007)*, Vienna, Austria, September 2007.

[17] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF", W3C Working Draft, 4 October 2006; http://www.w3.org/TR/2006/WD-rdf-sparql-query-20061004/.

[18] A. Patil, S. Oundhakar, A. Sheth, and K. Verma, "METEOR-S Web service Annotation Framework", *Proceedings of the World Wide Web Conference (WWW'04)*, July 2004; available at http://lsdis.cs.uga.edu/projects/meteor-s/.