



WSMO Deliverable
D11v0.2
**WSMO-LITE: LIGHTWEIGHT
SEMANTIC DESCRIPTIONS FOR
SERVICES ON THE WEB**

WSMO Working Draft – 21st December 2007

Authors:

Tomas Vitvar
Jacek Kopecký
Dieter Fensel

Editors:

Tomas Vitvar
Jacek Kopecký

This version:

<http://www.wsmo.org/TR/d11/v0.2/20071221/>

Latest version:

<http://www.wsmo.org/TR/d11/v0.2/>

Previous version:

<http://www.wsmo.org/TR/d11/v0.2/20071024/>



Abstract

In this deliverable we define a minimal set of semantic service descriptions in RDFS which can be used for annotations of various WSDL elements using SAWSDL annotation mechanism. We exploit the standard languages of W3C including RDF and RDFS as well as various extensions of those languages such as OWL, WSML and RIF for semantic service descriptions.



Contents

1	Introduction	4
2	Semantic Service Stack	4
2.1	Non-Semantic Level	5
2.2	Semantic Level	6
3	WSMO-Lite Service Ontology	7
4	WSMO-Lite Annotations for WSDL	8
5	Use of WSMO-Lite Annotations	12
6	Related work	13
7	Conclusion and Future Work	13



1 Introduction

Existing service specifications allow to describe service offerings for a client that can make an up-front decision on whether and how to consume the service's functionality. The most used specifications today are WSDL. Their uptake will further enable environments where thousands of services will have to be searched, integrated and mediated, and where automation will be the key enabler of service provisioning to end-users. In order to fulfill these challenges, *existing service specifications needs to be augmented with semantic descriptions*.

In 2007, the W3C finished its work on Semantic Annotations for WSDL and XML Schema (SAWSDL)¹. SAWSDL defines simple extensions for WSDL and XML Schema used to link WSDL components with arbitrary semantic descriptions. It thus provides the grounds for a bottom-up approach to service modeling: it supports the idea of adding small increments (and complexity) on top of WSDL, allowing to adopt results from various existing approaches. As the basis for bottom-up modeling, SAWSDL is independent of any particular semantic technology, i.e., it does not define any types, forms or languages for semantic descriptions.

In this paper we describe the WSMO-lite service ontology as the next evolutionary step after SAWSDL, filling the SAWSDL annotations with concrete semantic service descriptions and thus embodying the semantic layer of the Semantic Service Stack. With the ultimate goal to support realistic real-world challenges in intelligent service integration, WSMO-Lite addresses the following requirements:

- Identify the types and a simple vocabulary for semantic descriptions of services (a service ontology) as well as languages used to define these descriptions;
- Define an annotation mechanism for WSDL using this service ontology;
- Provide the bridge between WSDL, SAWSDL and (existing) domain-specific ontologies such as classification schemas, domain ontology models, etc.

The rest of this deliverable is structured as follows. In Section 2, we introduce the Semantic Service Stack along with the state-of-the-art technologies for services and Semantic Web languages used in the stack. In Section 3, we describe the WSMO-Lite Service Ontology and summarize the resolution of the major points from its development. In Section 4, we describe the WSMO-Lite semantic annotations for WSDL, and in Section 5 we describe the use of the WSMO-Lite annotations for some relevant services tasks.

2 Semantic Service Stack

As depicted in Figure 1, there are two levels in the Semantic Service Stack, namely *semantic* and *non-semantic* level. In addition, there are two types of stakeholders in the stack, namely a *service engineer* (human being) and a *client* (software agent). The service engineer uses Web services through the client, with particular tasks such as service discovery, selection, mediation, composition and invocation. Through these tasks the client or service engineer (depending on the level of automation) decide whether to bind with the service or not. In order to facilitate such decisions, services should describe their offers using so called *service contracts*. The Semantic Service Stack adopts the following general types of service contracts:

- *Information Model* defines the data model for input, output and fault messages.
- *Functional Descriptions* define service functionality, that is, what a service can offer to its clients when it is invoked.

¹<http://www.w3.org/TR/sawSDL/>

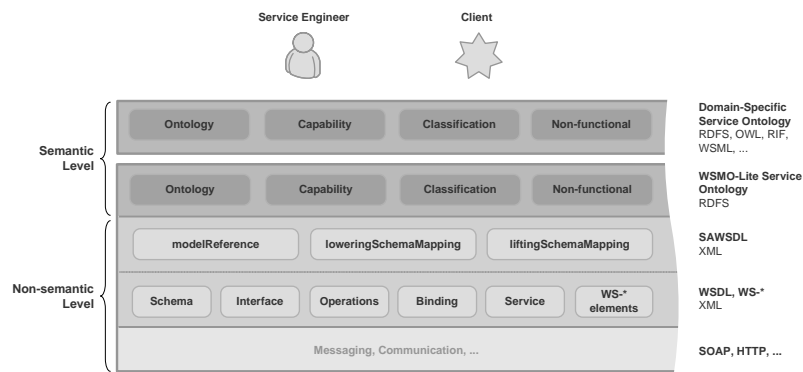


Figure 1: Semantic Service Stack

- *Non-Functional Descriptions* define any incidental details specific to the implementation or running environment of a service.
- *Behavioral Descriptions* define external (public choreography) and internal (private workflow) behavior.
- *Technical Descriptions* define messaging details, such as message serializations, communication protocols, and physical service access points.

In the following sections, we show how the Semantic Service Stack represents the above general description types for service contracts at the two different levels.

2.1 Non-Semantic Level

In regard to SOA technology developments today, the Semantic Service Stack represents service contracts at the non-semantic level using the existing de-facto and de-jure standards: WSDL, SAWSDL, and related WS-* specifications. They all use XML as a common flexible data exchange format. Service contracts are represented as follows:

- *Information Model* is represented using XML Schema.
- *Functional Description* is represented using a WSDL Interface and its operations.
- *Non-Functional Description* is represented using various WS-* specifications, such as WS-Policy, WS-Reliability, WS-Security, etc.
- *Behavioral Description* is represented using the WS-* specifications of WS-BPEL² (for the workflow) and WS-CDL³ (for the choreography).
- *Technical Description* is represented using WSDL Binding for message serializations and underlying communication protocols, such as SOAP, HTTP; and using WSDL Service for physical endpoint information.

In addition, while SAWSDL does not fall into any of the service contract descriptions, it is an essential part of the non-semantic level of the stack, providing the ground for the semantic layer. SAWSDL defines a simple extension layer that allows WSDL components to be annotated with semantics, using three extension attributes:

- *modelReference* for pointing to concepts that describe a WSDL component,
- *loweringSchemaMapping* and *liftingSchemaMapping* for specifying the mappings between the XML data and the semantic information model.

²<http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf>

³Choreography Description Language, <http://www.w3.org/TR/ws-cdl-10/>



2.2 Semantic Level

The Semantic Service Stack represents service contracts at the semantic level using the WSMO-Lite service ontology as follows (see Section 3 for a detailed description of WSMO-Lite):

- *Information Model* is represented using a domain ontology.
- *Functional Descriptions* are represented as *capabilities* and/or *functionality classifications*. A capability defines *conditions* which must hold in a state before a client can invoke the service, and *effects* which hold in a state after the service invocation. Classifications define the service functionality using some classification ontology (i.e., a hierarchy of categories).
- *Non-Functional Descriptions* are represented using an ontology, semantically representing some policy or other non-functional properties.
- *Behavioral Descriptions* are not represented explicitly in WSMO-Lite (see discussion in Section 3).
- *Technical Descriptions* are not represented semantically in the service ontology, as they are sufficiently covered by the non-semantic description in WSDL.

In order to create or reuse domain-specific service ontologies on top of the Semantic Service Stack, a service engineer can use any W3C-compliant language with an RDF syntax. This preserves the choice of language expressivity according to domain-specific requirements. Such languages may include RDF Schema (RDFS), Web Ontology Language (OWL) [4], Rule Interchange Format (RIF)⁴ or Web Service Modeling Language (WSML)⁵ [8].

RDF The W3C has produced several language recommendations for representation and exchange of knowledge on the Semantic Web. At the core, the Resource Description Framework (RDF)⁶ represents information in graph-based models with so called *triples*, i.e. statements in the form ⟨subject, predicate, object⟩. The subjects and objects link the triples into a graph. Thus, RDF can be used to represent the syntax of data using graph models while it does not define any semantics for any of the subjects, predicates and objects. RDF provides various serializations including RDF/XML⁷ and Notation 3 (N3)⁸.

RDFS On top of RDF, RDF Schema (RDFS)⁹ defines constructs that allow to express some semantics for the RDF model: RDFS allows to define classes describing the terminology of the domain of discourse, properties of those classes as well as class and property hierarchies (i.e. *subClassOf* and *subPropertyOf*). Thus, RDFS provides the minimal set of constructs that allow the specification of lightweight ontologies.

On top of RDFS: OWL, WSML, RIF Where the expressivity of the RDFS is not sufficient for modeling of the required knowledge, various specializations of RDFS can be used. Such specializations are being developed both inside and outside of W3C along the lines of knowledge representation paradigms of Description Logic (DL) and Logic Programming (LP).

The Web Ontology Language (OWL) [4] provides further vocabulary along with a formalism based on Description Logics. On the other hand, Web Service Modeling

⁴<http://www.w3.org/2005/rules/>

⁵For details on compliance of WSML with W3C languages see [2].

⁶<http://www.w3.org/RDF/>

⁷<http://www.w3.org/TR/rdf-syntax-grammar/>

⁸<http://www.w3.org/DesignIssues/Notation3.html>

⁹<http://www.w3.org/TR/rdf-schema/>



Language (WSML) defines several variants allowing for both paradigms of Description Logics (WSML-DL) and Logic Programming (WSML-Flight, WSML-Rule). All WSML variants can be represented using RDF syntax and they are layered on top of RDFS. While WSML-DL has a direct mapping to OWL, WSML-Rule is the basis of the Web Rule Language (WRL)¹⁰ specification which serves as an input for the W3C Rule Interchange Format Working Group (RIF WG)¹¹. RIF WG aims to produce a core rule language for the Semantic Web together with extensions that allow rules to be translated between different rule languages. The detailed description of WSML and its compliance with standards can be found in [2].

3 WSMO-Lite Service Ontology

Listing 1 shows the WSMO-Lite service ontology in RDFS, serialized in Notation 3. Below, we explain the semantics of the WSMO-Lite elements:

```

1 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
2 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
3 @prefix owl: <http://www.w3.org/2002/07/owl#> .
4 @prefix wl: <http://www.wsmo.org/ns/wsmo-lite#> .
5
6 wl:Ontology rdf:type rdfs:Class;
7   rdfs:subClassOf owl:Ontology.
8 wl:ClassificationRoot rdfs:subClassOf rdfs:Class.
9 wl:NonFunctionalParameter rdf:type rdfs:Class.
10 wl:Condition rdfs:subClassOf wl:Axiom.
11 wl:Effect rdfs:subClassOf wl:Axiom.
12 wl:Axiom rdf:type rdfs:Class.

```

Listing 1: WSMO-Lite Service Ontology

- *wl:Ontology* (lines 6–7) defines a container for a collection of assertions about the information model of a service. Same as *owl:Ontology*, *wl:Ontology* allows for meta-data such as comments, version control and inclusion of other ontologies. *wl:Ontology* is a subclass of *owl:Ontology* since as we already mentioned, it has a special meaning of the ontology used as the service information model.
- *wl:ClassificationRoot* (line 8) marks a class that is a root of a classification which also includes all the RDFS subclasses of the root class. A classification (taxonomy) of service functionalities can be used for functional description of a service.
- *wl:NonFunctionalParameter* (line 9) specifies a placeholder for a concrete domain-specific non-functional property.
- *wl:Condition* and *wl:Effect* (lines 10–12) together form a *capability* in functional service description. They are both subclasses of a general *wl:Axiom* class through which a concrete language can be used to describe the logical expressions for conditions and effects. We illustrate this on an example in Listing 2 (lines 26–42).

Below, we discuss some relevant issues related to WSMO-Lite Service Ontology.

1. Relation of WSMO-Lite to WSMO. WSMO-Lite has been created due to a need for lightweight service ontology which would directly build on the newest W3C standards and allow bottom-up modeling of services. On the other hand, WSMO is an established framework for Semantic Web Services representing a top-down model identifying semantics useful in a semantics-first environment. WSMO-Lite adopts the WSMO model and makes its semantics lighter in the following major aspects:

¹⁰<http://www.w3.org/Submission/2005/08/>

¹¹<http://www.w3.org/2005/rules/>



- WSMO defines formal user goals and mediators, while WSMO-Lite treats mediators as infrastructure elements, and specifications for user goals as dependent on the particular discovery mechanism used. They both can be adopted in the running environment in combination with WSMO-Lite.
- WSMO-Lite only defines semantics for the information model, functional and non-functional descriptions (as WSMO Service does) and only implicit behavior semantics (see below). If needed, an application can extend WSMO-Lite with its own explicit behavioral descriptions, or it can adopt other existing technologies.
- While WSMO uses the WSML language for describing domain-specific semantic models, WSMO-Lite allows the use of any ontology language with an RDF syntax (see Section 2.2 for more details).

2. WSMO-Lite does not define explicit behavioral descriptions. WSMO-Lite treats Web services as atomic, and it does not deal with the internals of services. For this reason, it contains no construct for describing a private behavior (internal workflow) nor does it deal with annotations of existing WS-BPEL processes. Semantic annotation of processes is an independent research effort led by the business process community¹² and its use in combination with WSMO-Lite services is an open research question. Nevertheless, WSMO-Lite allows to derive behavioral descriptions, defined according to WSMO choreography [9], from functional (capability) annotations of services.

3. Dependency of WSMO-Lite on SAWSDL. As we already mentioned, WSMO-Lite has been created to address the need for a concrete service ontology as the next evolutionary step after SAWSDL. For this reason it might seem that WSMO-Lite is also SAWSDL-dependent. However, WSMO-Lite uses SAWSDL only as an annotation mechanism for WSDL (see Section 4) while the WSMO-Lite service ontology can be used with any machine-readable service descriptions in combination with an appropriate annotation mechanism.

4. WSMO-Lite annotations for RESTful services. In this paper we define how WSMO-Lite can be used on top of WSDL and SAWSDL. RESTful services (i.e., resource-oriented services using plain HTTP as the communication protocol) can also be described in WSDL, which has support for describing Web services that do not use SOAP as the communication protocol. Such descriptions can also be annotated using WSMO-Lite. On the other hand, there are efforts in defining an annotation mechanism for RESTful services without WSDL, such as SA-REST [10]. This work is complementary to WSMO-Lite and we expect to integrate it in the W3C SWS-Testbed XG¹³.

5. Concrete semantics for conditions and effects. To work with conditions and effects, it is necessary to define the environment in which these axioms are evaluated. Such an environment depends on the particular logical language in which the axioms are expressed. WSMO-Lite does not prescribe any concrete language for functional service semantics, and therefore it cannot define semantics for conditions and effects as they are language-dependent.

4 WSMO-Lite Annotations for WSDL

Section 2.2 mentions briefly how the WSMO-Lite ontology is used for the semantic descriptions. In this section, we define the particular types of annotations supported

¹²More details on this issue can be found for instance at <http://www.ip-super.org>

¹³<http://www.w3.org/2005/Incubator/swsc/>

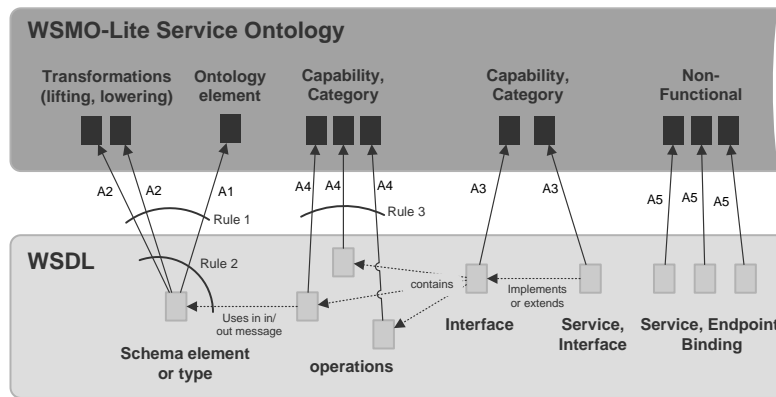


Figure 2: Illustration of Annotations and Rules

by WSMO-Lite. For this purpose we define two types of annotations, namely *reference annotations* and *transformation annotations*. A reference annotation points from a WSDL component (XML Schema message or type definition, interface, operation, binding, service, or endpoint) to a WSMO-Lite semantic concept. SAWSDL represents this type of annotation using *modelReference* extension attribute. A transformation annotation specifies a data transformation called *lifting* from a component of XML schema to an element of ontology; and a reverse transformation (from ontology to XML) called *lowering*. SAWSDL represents this annotation using extension attributes *loweringSchemaMapping* and *liftingSchemaMapping* respectively.

Figure 2 illustrates a set of annotations (marked *A1...A5*) and their associated rules (marked *Rule 1...Rule 3*). The purpose of the rules is to ensure that the annotations are:

- *complete*, that is, no gaps are left in the semantic annotations, so that the client can see all the parts of the service description; for instance, all the operations should be semantically annotated so that they are reachable to automatic discovery.
- *consistent*, that is, no related annotations are contradictory; for instance the schema annotations by model reference need to point to concepts that are the outputs of the lifting schema mapping transformation, or inputs of the lowering one.

Listing 2 shows an example ontology we use to illustrate annotations. It defines a simple ontology for a telecommunication service (lines 9–24); the capability for a concrete Video on Demand subscription service (lines 26–39) (the condition says that the customer must have a network connection with some minimal bandwidth, the effect says that the customer is subscribed to the service); a non-functional property describing the pricing (lines 44–48); and a simple functionality classification (lines 50–53). We also define the *wsm:Literal* data type (line 42) for WSMO-Lite axioms so that a client can correctly process them according to the WSMO specification.

A1: Annotations of XML Schema (ontology). The schema used in WSDL to describe messages, i.e., the element declarations and type definitions, can carry reference annotations linking to classes from the service information model ontology.

A2: Annotations of XML Schema (transformations). To be able to communicate with a service, the client needs to transform data between its semantic model and the service-specific XML message structures. The schema may contain transformation annotations (lifting or lowering) which specify the appropriate mappings.

```

1 <xs:element name="NetworkConnection" type="NetworkConnectionType"
2   sawsdl:modelReference="http://example.org/onto#NetworkConnection"
3   sawsdl:loweringSchemaMapping="http://example.org/NetCn.xslt"/>

```



```

1 // namespaces and prefixes
2 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
3 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
4 @prefix wl: <http://www.wsmo.org/ns/wsmo-lite#> .
5 @prefix ex: <http://example.org/onto#> .
6 @prefix xs: <http://www.w3.org/2001/XMLSchema#> .
7 @prefix wsml: <http://www.wsmo.org/wsml/wsml-syntax#> .
8
9 // ontology example
10 <> rdf:type wl:Ontology.
11
12 ex:Customer rdf:type rdfs:Class .
13 ex:hasService rdf:type rdf:Property ;
14   rdfs:domain ex:Customer ;
15   rdfs:range ex:Service .
16 ex:Service rdf:type rdfs:Class .
17 ex:hasConnection rdf:type rdf:Property ;
18   rdfs:domain ex:Customer ;
19   rdfs:range ex:NetworkConnection .
20 ex:NetworkConnection rdf:type rdfs:Class .
21 ex:providesBandwidth rdf:type rdf:Property ;
22   rdfs:domain ex:NetworkConnection ;
23   rdfs:range xs:integer .
24 ex:VideoOnDemandService rdfs:subClassOf ex:Service .
25
26 // capability description example
27 ex:VideoOnDemandSubscriptionPrecondition rdf:type wl:Condition ;
28   rdf:value "
29     ?customer[hasConnection hasValue ?connection]
30     memberOf Customer and
31     ?connection[providesBandwidth hasValue ?y]
32     memberOf NetworkConnection and
33     ?y > 1000
34   ""^wsml:AxiomLiteral .
35
36 ex:VideoOnDemandSubscriptionEffect rdf:type wl:Effect ;
37   rdf:value "
38     ?customer[hasService hasValue ?service]
39   ""^wsml:AxiomLiteral .
40
41 // definition of the axiom for WSML language
42 wsml:AxiomLiteral rdf:type rdfs:Datatype .
43
44 // non-functional property example
45 ex:PriceSpecification rdfs:subClassOf wl:NonFunctionalParameter .
46 ex:VideoOnDemandPrice rdf:type ex:PriceSpecification ;
47   ex:pricePerChange "30"^^ex:euroAmount ;
48   ex:installationPrice "49"^^ex:euroAmount .
49
50 // classification example
51 ex:SubscriptionService rdf:type wl:ClassificationRoot .
52 ex:VideoSubscriptionService rdfs:subClassOf ex:SubscriptionService .
53 ex:NewsSubscriptionService rdfs:subClassOf ex:SubscriptionService .

```

Listing 2: Example of domain-specific service ontology

Listing 3: Example of annotations A1 and A2

Listing 3 shows an example of annotations *A1* and *A2* (the lowering transformation is omitted for brevity). Below, we define Rule 1 for consistency of *A1* and *A2* annotations on schema components; and Rule 2 for completeness of these annotations on element declarations used as operation input and output messages.

Rule 1 (consistency): If a XML schema message has *A1* and *A2* annotations then a semantic concept identified by *A1* must be possible to transform (lower and lift defined by *A2*) to/from the XML Schema message.

Rule 2 (completeness): Every input message element of WSDL interface must have consistent *A1* annotation with *A2* annotation (lowering). Every output message element must have consistent *A1* annotation with *A2* annotation (lifting).



A3: Annotations of WSDL Interface and Service (functional). Functional descriptions (both capabilities and categories) apply both to concrete web services and to the reusable and abstract interfaces. A reference annotation points from a service or an interface to its appropriate functional description. Listing 4 shows an example of multiple A3 annotations:

```

1 <wsdl:interface name="NetworkSubscription"
2     sawsdl:modelReference="http://example.org/onto#VideoSubscriptionService
3     http://example.org/onto#VideoOnDemandSubscriptionPrecondition
4     http://example.org/onto#VideoOnDemandSubscriptionEffect" >
```

Listing 4: Example of annotations A3

Please note that a WSDL interface may be shared by multiple services, therefore the functional description of the interface should be general. A concrete functional description attached to the service then refines the functional description of the interface. Additionally, aggregate interfaces or services (i.e., those that combine multiple potentially independent functionalities) may be annotated with multiple functional descriptions.

A4: Annotations of WSDL Interface operations (functional). Functional descriptions (both capabilities and categories) apply also to interface operations, to indicate their particular functionalities. A reference annotation points from an operation to its appropriate functional description.

Functional annotation of interface operations can be used for services whose interfaces are simply collections of standalone operations. For example, a network subscription service may offer independent operations for subscription to a bundle, cancellation of a subscription, or price inquiry. A client will generally only want to use one or two of these operations, not all three. This shows that service discovery can, in such cases, become operation discovery. Also, operation annotations can be used for defining the order in which the operations should be invoked. Rule 3 defines completeness for A4 annotations:

Rule 3 (completeness): All operations within an interface must be annotated with a functional description. This rule ensures that no operation is left invisible to the automated clients.

Please note that annotations A3 and A4 apply to both types of functional descriptions, i.e., a capability or a category from some functional classification. It is even possible to combine them for a service, interface and its operations.

A5: Annotations of WSDL Service, Endpoints, and Binding (non-functional). Non-functional descriptions apply to a concrete instance of a Web service, that is, a Service, its Endpoints, or its Binding. A reference annotation can point from any of these components to a non-functional property. Listing 5 shows an example of annotation A5:

```

1 <wsdl:service name="ExampleCommLtd"
2     interface="NetworkSubscription"
3     sawsdl:modelReference="http://example.org/onto#VideoOnDemandPrice">
4     <wsdl:endpoint ...
5 </wsdl:service>
```

Listing 5: Example of annotation A5

Please note that non-functional descriptions are always specific to a concrete service, therefore, annotating interfaces or interface operations with non-functional properties is not defined. In case non-functional properties need to be specified on the operations (for example, different operations may have different invocation micropayment prices), a WSDL binding operation components (which mirror the operations of some interface) may be used to capture these properties.



5 Use of WSMO-Lite Annotations

In this section, we discuss the use of the different types of WSMO-Lite annotations for automation of the various tasks that the client may perform with services. Not all annotations described in Section 4 are always needed, only those required by the tasks at hand in a particular domain-specific setting. Table 1 provides a summary, with $A1 \dots A5$ denoting the annotations and $R1 \dots R3$ denoting the rules. The symbol \bullet marks the annotations and rules required to automate a given task, and the symbol \circ marks rules that are helpful but not absolutely required.

Service Task	A1	A2	A3	A4	A5	R1	R2	R3
Service Discovery			\bullet					
Operation Discovery				\bullet				\circ
Composition			\bullet					
Ranking and Selection					\bullet			
Operation Invocation	\bullet	\bullet				\bullet	\bullet	
Service Invocation	\bullet	\bullet		\bullet		\bullet	\bullet	\circ
Data Mediation	\bullet	\bullet				\bullet		
Process Mediation	\bullet	\bullet		\bullet		\bullet	\bullet	\circ

Table 1: Service Tasks, Annotations and Rules

- *Service Discovery*, operating on functional descriptions (capabilities or categories), requires annotations $A3$.
- *Operation Discovery*, operating on functional descriptions of individual operations, requires annotations $A4$. Operation discovery might be useful with interfaces that are collections of standalone, independent operations. Rule 3 ensures that no operation is left invisible to this discovery process.
- *Composition* uses capability descriptions, i.e., annotations $A3$ restricted to capabilities, to put together multiple services to achieve a complex goal.
- *Ranking and Selection* processes non-functional descriptions, i.e., annotations $A5$, to select the service that most suits some particular requirements.
- *Operation Invocation* is the invocation of a single operation, requiring data transformations between the semantic model on the client and the service's XML message structure. This requires $A1$ and $A2$ annotation, kept consistent by Rule 1. Rule 2 ensures that all operation messages have these annotations.
- *Service Invocation* requires the operations of the service to be invoked in a proper order. This task therefore requires annotations $A4$. Rule 3 ensures that no operation is omitted from the choreography.
- *Data Mediation* uses data annotations ($A1$ and $A2$) — assuming two different schemas correspond to a single shared ontology, the $A1$ annotations make it possible to discover such a correspondence, and the $A2$ annotations then enable data mapping transformations: lifting from one schema and lowering to the other.
- *Process Mediation* combines data mediation and choreography processing and thus requires the combined annotations $A1$, $A2$ and $A4$. As described in [3], process mediation is applied during conversation between two services mediating their choreographies and messages.

This provides certain modularity to WSMO-Lite, enabling different environments using this service ontology to mix and match the annotations as necessary for the required tasks. On top of already being light-weight, WSMO-Lite provides value even if only parts of it are used.



6 Related work

The major stream of related work is in the frameworks for Semantic Web Services (SWS), including WSMO [8], Semantic Markup for Web Services (OWL-S [11]) and Web Service Semantics (WSDL-S [1]). WSMO is a top-down conceptual model for SWS that defines four top-level components: ontologies, mediators, goals and web services. As we already mentioned, WSMO was the major input for WSMO-Lite. On the other hand, OWL-S was the first major ontology for SWS defining three inter-linked ontologies: Service Profile (for the functional and non-functional descriptions), Service Model (for the behavioral descriptions), and Service Grounding (for physical Web service access). There are also recent works on OWL-S grounding that uses SAWSDL [7, 6]. In comparison with that work, WSMO-Lite takes the additional step of simplifying the annotations into a lightweight ontology. WSDL-S was created in the METEOR-S¹⁴ project as a specification of how WSDL can be annotated with semantic information. WSDL-S itself does not provide a concrete model for SWS, instead it makes the assumption that the concrete model will be expressible as annotations in WSDL and XML Schema documents. WSDL-S was taken as the basis for SAWSDL. In addition, there is a major orthogonal work to WSMO-Lite called SA-REST [10], aiming to enrich the informal descriptions of RESTful services, usually available in HTML, with RDFa¹⁵ annotations. This work is complementary to WSMO-Lite as it could serve as an additional annotation mechanism for WSMO-Lite service ontology used for RESTful services. We will work on such integration withing the W3C SWS-Testbed XG.

7 Conclusion and Future Work

In this paper, we describe the latest results from the development of WSMO-Lite, a minimal lightweight ontology for Semantic Web Services, building on the newest W3C standards. WSMO-Lite fills in SAWSDL annotations, and thus enables the Semantic Service Stack, open for various customizations according to domain-specific requirements, languages of required expressivity and domain-specific ontologies. WSMO-Lite supports the idea of incremental enhancements of SAWSDL as Amit Sheth points out in [5]: “Rather than look for a clear winner among various SWS approaches, I believe that in the post-SAWSDL context, significant contributions by each of the major approaches will likely influence how we incrementally enhance SAWSDL. Incrementally adding features (and hence complexity) when it makes sense, by borrowing from approaches offered by various researchers, will raise the chance that SAWSDL can present itself as the primary option for using semantics for real-world and industry-strength challenges involving Web services.”

In our future work we plan to work on validation of WSMO-Lite annotations, together with a compiler for WSMO-Lite descriptions. We also plan to integrate WSMO-Lite with other research efforts within the W3C SWS-Testbed XG (such as already mentioned SA-REST) and to support service mashups with the WSMO-Lite ontology. In addition, we plan to integrate the WSMO-Lite ontology with the results of the semantic business processes research.

References

- [1] R. Akkiraju, J. Farrell, J. Miller, M. Nagarajan, M. M. Schmidt, Amit Sheth, and Kunal Verma. Web Service Semantics - WSDL-S, available at

¹⁴<http://lsdis.cs.uga.edu/projects/meteor-s/>

¹⁵<http://www.w3.org/TR/xhtml1-rdfa-primer/>



- <http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/>. Technical report, LSDIS Lab, 2005.
- [2] Jos de Bruijn, Dieter Fensel, and Holger Lausen. D34v0.1: The Web Compliance of WSML. Technical report, DERI, 2007. Available from: <http://www.wsmo.org/TR/d34/v0.1/>.
- [3] Thomas Haselwanter, Paavo Kotinurmi, Matthew Moran, Tomáš Vitvar, and Maciej Zaremba. Wsmx: A semantic service oriented middleware for b2b integration. In *ICSOC*, pages 477–483, 2006.
- [4] Ian Horrocks. Owl: A description logic based ontology language. In *CP*, pages 5–8, 2005.
- [5] David Martin and John Domingue. Semantic web services: Past, present and possible futures (systems trends and controversies). *IEEE Intelligent Systems*, 22(6), 2007.
- [6] David Martin, Massimo Paolucci, and Matthias Wagner. Bringing Semantic Annotations to Web Services: OWL-S from the SAWSDL Perspective. In Karl Aberer, Key-Sun Choi, Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web*, volume 4825 of *LNCS*, pages 340–352. Springer, 2007.
- [7] Massimo Paolucci, Matthias Wagner, and David Martin. Grounding OWL-S in SAWSDL. In Bernd J. Krämer, Kwei-Jay Lin, and Priya Narasimhan, editors, *ICSOC*, volume 4749 of *Lecture Notes in Computer Science*, pages 416–421. Springer, 2007.
- [8] D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web Service Modeling Ontology. *Applied Ontology*, 1(1):77–106, 2005.
- [9] Dumitru Roman and James Scicluna. Ontology-based choreography of wsmo services. Wsmo d14 final draft v0.3, DERI, 2006. Available at: <http://www.wsmo.org/TR/d14/v0.3/>.
- [10] Amit P. Sheth, Karthik Gomadam, and Jon Lathem. SA-REST: Semantically Interoperable and Easier-to-Use Services and Mashups. *IEEE Internet Computing*, 11(6):91–94, 2007.
- [11] The OWL Services Coalition. OWL-S 1.1 Release. Available at <http://www.daml.org/services/owl-s/1.1/>, November 2004.