Project Number:	**215219**
Project Acronym:	**SOA4All**
Project Title:	**Service Oriented Architectures for All**
Instrument:	**Integrated Project**
Thematic Priority:	**Information and Communication Technologies**

D3.4.2
WSMO-Lite: Lightweight Semantic Descriptions for Services on the Web

Activity:	Activity 2 — Core R&D Activities	
Work Package:	WP3 — Service Annotation and Reasoning	
Due Date:		M12
Submission Date:		10/3/2009
Start Date of Project:		01/03/2008
Duration of Project:		36 Months
Organisation Responsible for Deliverable:		UIBK
Revision:		
Author(s):	Jacek Kopecký	UIBK
	Tomas Vitvar	UIBK
	Dieter Fensel	UIBK
Reviewer(s):	Carlos Pedrinaci	OU
	Rafa Cabero	ATOS
	Jean-Pierre Lorre	EBM

Project co-funded by the European Commission within the Seventh Framework Programme (2007–2013)		
Dissemination Level		
PU	Public	☒

Version History

Version	Date	Comments, Changes, Status	Authors, Contributors, Reviewers
0.9	3/2/2009	CMS WG draft	All authors
1.0	8/3/2009	Final version after internal reviews	Jacek Kopecky (UIBK), Carlos Pedrinaci (OU), Rafa Cabero (ATOS), Jean-Pierre Lorre (EBM)

TABLE OF CONTENTS

List of Figures

List of Tables

Glossary of Acronyms

Acronym	**Definition**
CMS WG	Conceptual Models for Services Working Group
DL	Description Logics
D	Deliverable
EC	European Commission
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
LP	Logic Programming
NFP	Non-Functional Property
OWL	Web Ontology Language
RDFS	RDF Schema
RDF	Resource Description Framework
RIF	Rule Interchange Format
SAWSDL	Semantic Annotations for WSDL and XML Schema
SEE	Semantic Execution Environment
SOA	Service-Oriented Architecture
SWS	Semantic Web Services
WG	Working Group
WP	Work Package
WRL	Web Rule Language
WS-BPEL	Business Process Execution Language for Web Services
WS-CDL	Web Services Choreography Description Language
WSDL	Web Services Description Language
WSML	Web Service Modeling Language
WSMO	Web Service Modeling Ontology
WSMX	Web Service Execution Environment
XML	Extensible Markup Language

1 EXECUTIVE SUMMARY

In this deliverable we define WSMO-Lite, a lightweight set of semantic service descriptions in RDFS that can be used for annotations of various WSDL elements using the 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.

In particular, we distinguish four kinds of semantics of services:

- functional - what the service does;
- behavioral - how to interact with the service;
- information model - the meaning of the information exchanged with the service;
- nonfunctional - other properties.

We define RDFS classes for marking ontology elements that express functional, nonfunctional and information model semantics; these models are attached to the service and its message descriptions. To describe the behavioral semantics, we attach functional descriptions to the service's operations.

We also define a minimal RDF model for expressing the information present in WSMO-Lite service description documents, which will be used in SOA4All service registry.

2 INTRODUCTION

Existing service specifications allow one to describe service offerings so that a client can make an up-front decision on whether and how to consume the service's functionality. Most of the specifications used today are expressed in 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 need to be augmented with semantic descriptions [16].

In 2007, the W3C finished its work on Semantic Annotations for WSDL and XML Schema (SAWSDL [14]). 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 semantic service modeling: it supports the idea of adding small increments (and complexity) on top of WSDL, allowing results from various existing approaches to be adopted. 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 deliverable, 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 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.

Even though we adopt the base Web service model from WSDL and SAWSDL, WSMO-Lite is inspired in the WSMO ontology [12], however, only focusing on a subset of it using it to define a gradual extension of SAWSDL.

2.1 Alignment with SOA4All Architecture and Use Cases

WSMO-Lite is the description language that will be used in SOA4All for semantic descriptions of Web services. These descriptions will be stored in a service registry, represented in RDF according to the model defined in Chapter 6, and used by the semantic automation components.

The use cases deal with Web services, both WSDL-based and RESTful. WSMO-Lite will be used by the use cases to describe the WSDL-based services. Additionally, the WSMO-Lite service ontology from Chapter 4 and the minimal RDF representation from Chapter 6 are also used by MicroWSMO, the language for describing RESTful services.

In effect, WSMO-Lite will be used by all the use cases, whether directly, or through MicroWSMO.

2.2 Structure of the deliverable

The rest of this deliverable is structured as follows. In Chapter 3, 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 Chapter 4, we describe the WSMO-Lite Service Ontology and summarize the resolution of the major points from its development, including details on the relationship to WSMO and WSML. In Chapter 5, we describe the WSMO-Lite semantic annotations for WSDL, and in Chapter 6 we define a minimal RDF representation for WSMO-Lite service descriptions, which may be useful for service registries. In Chapter 7, we show how we can map a WSDL document annotated with WSMO-Lite semantics into a WSML description. Finally, Chapter 8 discusses related work, and Chapter 9 concludes the deliverable and discusses some tasks left for the future.

3 Semantic Service Stack

As depicted in Figure 3.1, 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. In order to facilitate some degree of automation in these tasks, services should describe their offers in a machine-readable form using so called *service contracts*. The Semantic Service Stack adopts the following general types of service contracts (adapted from [15]):

- *Information Model* defines the data model for input, output and fault messages, as well as for the data relevant to the other aspects of the service description.
- *Functional Descriptions* define service functionality, that is, what a service can offer to its clients when it is invoked.
- *Nonfunctional Descriptions* define any incidental details specific to a service provider or to the service implementation or its running environment. An example nonfunctional property is the price; the functionality of a service is generally not affected by the price, even though the desirability may be. Nonfunctional properties also include Quality of Service (QoS) aspects such as performance, reliability and so on.
- *Behavioral Descriptions* define external and internal behavior. The former is the description of a public *choreography*, the protocol that a client needs to follow when consuming a service's functionality[1]; and the latter is a description of a workflow, i.e., how the functionality of the service is aggregated out of other services.
- *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.

3.1 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 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.
- *Nonfunctional Description* is represented using various WS-* specifications (WS-Policy [23], WS-Agreement [20], WS-Reliability [24], WS-Security [6], etc.).

[1]This view of choreography is adopted from WSMO, it is somewhat different from how choreography is defined in WS-CDL [22].

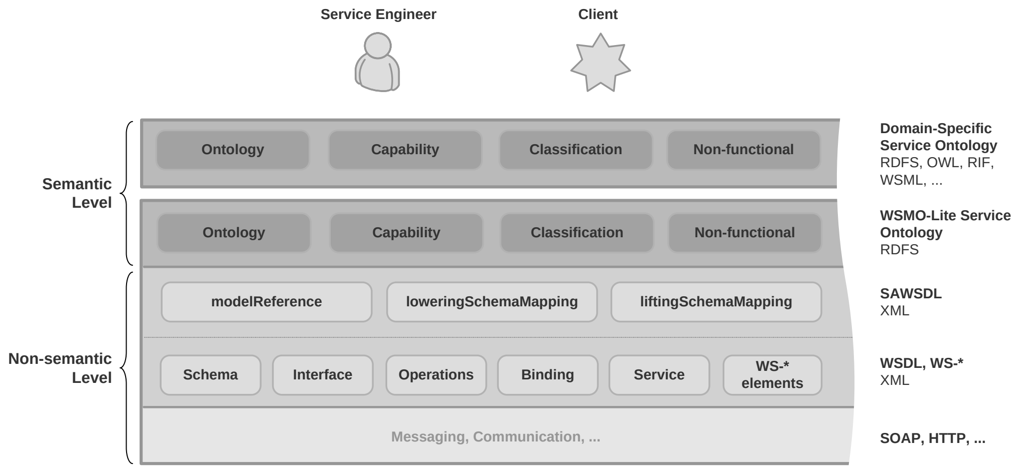

Figure 3.1: Semantic Service Stack

- *Behavioral Description* is represented using the relevant WS-* specifications, such as WS-BPEL [21] (for the workflow) and WS-CDL [22] (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 groundwork 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 semantic concepts that describe a WSDL component,
- *loweringSchemaMapping* and *liftingSchemaMapping* for specifying the mappings between the XML data and the semantic information model.

3.2 Semantic Level

In the Semantic Service Stack, we represent service contracts at the semantic level using the WSMO-Lite service ontology as follows (see Chapter 4 for a detailed description of WSMO-Lite):

- *Information Model* is represented using a domain ontology, along with associated data lifting and lowering transformations.
- *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*).
- *Nonfunctional Descriptions* are represented using an ontology, semantically representing some policy or other nonfunctional properties.

- *Behavioral Descriptions* are not represented explicitly in WSMO-Lite (see discussion in Chapter 4).
- *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 [2]), Rule Interchange Format (RIF)[2] or Web Service Modeling Language (WSML [18]).

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 [10]) 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/XML [9] and Notation 3 (N3).[3]

RDFS On top of RDF, RDF Schema (RDFS [11]) defines constructs that allow the expression of some semantics for the RDF model: RDFS allows the definition of 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 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) [2] provides a vocabulary along with a formalism based on Description Logics. The Web Service Modeling 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)[4] specification which serves as an input for the W3C Rule Interchange Format Working Group (RIF WG). The working group aims to produce a core rule language for the Semantic Web together with extensions that allow rules to be translated between different rule languages.

[2]Work in progress at the Rule Interchange Format WG `http://www.w3.org/2005/rules/`

[3]`http://www.w3.org/DesignIssues/Notation3.html`

[4]`http://www.w3.org/Submission/2005/08/`

4 WSMO-Lite Service Ontology

Listing 4.1 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 wsl: <http://www.wsmo.org/ns/wsmo−lite#> .
5
6  wsl:Ontology a rdfs:Class;
7    rdfs:subClassOf owl:Ontology.
8  wsl:FunctionalClassificationRoot rdfs:subClassOf rdfs:Class.
9  wsl:NonFunctionalParameter a rdfs:Class.
10 wsl:Condition a rdfs:Class.
11 wsl:Effect a rdfs:Class.
```

Listing 4.1: WSMO-Lite Service Ontology

- *wsl:Ontology* (lines 6–7) defines a container for a collection of assertions about the information model of a service. *wsl:Ontology* is a subclass of *owl:Ontology* limited to such ontologies that may serve as information models. OWL ontology meta-data such as comments, version control and inclusion of other ontologies, are also allowed on *wsl:Ontology*.
- *wsl:FunctionalClassificationRoot* (line 8) marks a class that is a root of a classification which also includes all the RDFS subclasses of the root class (the actual functional categories). A classification (taxonomy) of service functionalities can be used for functional description of a service.
- *wsl:NonFunctionalParameter* (line 9) specifies a placeholder for a concrete domain-specific nonfunctional property.
- *wsl:Condition* and *wsl:Effect* (lines 10–12) together form a *capability* in a functional service description. Both are expected to use a concrete logical language to describe the logical expressions for conditions and effects. We illustrate this on an example in Listing 5.1 (lines 26–42).

In the remainder of this chapter, 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 a lightweight service ontology which directly builds on the newest W3C standards and allows 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 adapts the WSMO model and makes its semantics lighter in the following major aspects:

- Beside Web services, WSMO also defines goals and mediators; these are currently not present in WSMO-Lite which is geared towards only describing services through ontologies.

- Where WSMO expects the functional capabilities of services to be described using logical expressions as preconditions, assumptions, postconditions and effects, WSMO-Lite only distinguishes between preconditions and effects (which may include assumptions or postconditions, respectively). WSMO-Lite further specifies a coarser-grained mechanism to describe functionality using functionality classifications (taxonomies), for which WSMO does not have direct support.
- Compared to WSMO Web service descriptions, WSMO-Lite only defines semantics for the information model, functional and nonfunctional descriptions and only implicit behavioral description (see below). If needed, an application can extend WSMO-Lite with its own explicit behavioral descriptions, or it can adopt other existing technologies.

Chapter 7 shows a mapping from WSMO-Lite descriptions into WSML.

2. WSMO-Lite defines behavioral descriptions through functional annotations of operations. While WSMO-Lite does not have a special construct for choreography descriptions, the behavioral aspects are inferred from functional (capability) annotations of individual service operations. Such annotations can be transformed into a WSMO choreography [13], using the algorithm described in [19].

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 Chapter 5) while the WSMO-Lite service ontology can be used with any machine-readable service descriptions in combination with an appropriate annotation mechanism.

4. 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. An application that uses a concrete language for expressing conditions and effects is also responsible for defining how the conditions and effects are processed.

5 WSMO-LITE ANNOTATIONS FOR WSDL

Section 3.2 mentions briefly how the WSMO-Lite ontology is used for the semantic descriptions. In this chapter, we define the particular types of annotations supported 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 element declaration or type definition, WSDL interface, operation, service) 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.[1]

Listing 5.1 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 nonfunctional property describing the pricing (lines 44–48); and a simple functionality classification with three categories (lines 50–53). We also define the *wsml:AxiomLiteral* data type (line 42) for WSML-Flight logical expressions so that a client can correctly process them according to the WSML specification.

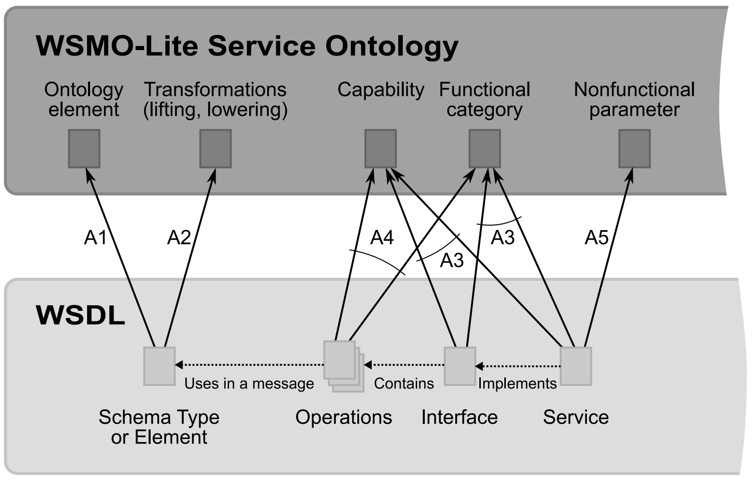

Figure 5.1: Illustration of WSMO-Lite Annotations

Figure 5.1 illustrates the positioning of WSMO-Lite annotations (marked *A1...A5*), which are defined as follows:

Annotation A1: Ontological annotations of XML Schema. 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.

Annotation A2: Transformation annotations of XML Schema. To be able to communicate with a service, the client needs to transform data between its semantic model and the

[1]The actual expression of concrete lifting or lowering transformations is out of scope of this deliverable.

```
# namespaces and prefixes
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix wsl: <http://www.wsmo.org/ns/wsmo-lite#> .
@prefix ex: <http://example.org/onto#> .
@prefix xs: <http://www.w3.org/2001/XMLSchema#> .
@prefix wsml: <http://www.wsmo.org/wsml/wsml-syntax#> .

# ontology example
<> a wsl:Ontology.

ex:Customer a rdfs:Class .
ex:hasService a rdf:Property ;
    rdfs:domain ex:Customer ;
    rdfs:range ex:Service .
ex:Service a rdfs:Class .
ex:hasConnection a rdf:Property ;
    rdfs:domain ex:Customer ;
    rdfs:range ex:NetworkConnection .
ex:NetworkConnection a rdfs:Class .
ex:providesBandwidth a rdf:Property ;
    rdfs:domain ex:NetworkConnection ;
    rdfs:range xs:integer .
ex:VideoOnDemandService rdfs:subClassOf ex:Service .

# capability description example
ex:VideoOnDemandSubscriptionPrecondition a wsl:Condition ;
    rdf:value ”””
        ?customer[ex#hasConnection hasValue ?connection]
            memberOf ex#Customer and
        ?connection[ex#providesBandwidth hasValue ?y]
            memberOf ex#NetworkConnection and
        ?y > 1000
    ”””^^wsml:AxiomLiteral .
ex:VideoOnDemandSubscriptionEffect a wsl:Effect ;
    rdf:value ”””
        ?customer[ex#hasService hasValue ?service]
            memberOf ex#Customer and
        ?service memberOf VideoOnDemandSubscription
    ”””^^wsml:AxiomLiteral .

# definition of the axiom for WSML language
wsml:AxiomLiteral a rdfs:Datatype .

# nonfunctional property example
ex:PriceSpecification rdfs:subClassOf wsl:NonFunctionalParameter .
ex:VideoOnDemandPrice a ex:PriceSpecification ;
   ex:pricePerChange ”30”^^ex:euroAmount ;
   ex:installationPrice ”49”^^ex:euroAmount .

# classification example
ex:SubscriptionService a wsl:FunctionalClassificationRoot .
ex:VideoSubscriptionService rdfs:subClassOf ex:SubscriptionService .
ex:NewsSubscriptionService rdfs:subClassOf ex:SubscriptionService .
```

Listing 5.1: Example of domain-specific service ontology

service-specific XML message structures. The schema may contain transformation annotations (lifting or lowering) which specify the appropriate mappings.

Listing 5.2 shows an example of annotations *A1* and *A2* (the content of the lowering transformation is omitted for brevity).

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

Listing 5.2: Example of annotations A1 and A2

Annotation A3: Functional annotations of WSDL Interface and Service. 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 5.3 shows an example of multiple *A3* annotations.

```
<wsdl:interface name="NetworkSubscription"
        sawsdl:modelReference="http://example.org/onto#VideoSubscriptionService
                                         http://example.org/onto#VideoOnDemandSubscriptionPrecondition
                                         http://example.org/onto#VideoOnDemandSubscriptionEffect" >
   <wsdl:operation name="CheckNetworkConnection" ... />
</wsdl:interface>
```

Listing 5.3: 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.

Annotation A4: Functional annotations of WSDL Interface operations. 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 functionality has different, separate sub-parts. For example, a network subscription service may offer 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. Service invocation will therefore need to select only the operations applicable to the current goal, and invoke them in the correct order, which is also partially implied by the functional operation annotations.

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 capabilities and classification together. However, WSMO-Lite does not define any formal relationship between capabilities and classifications on the same component, or between annotations *A3* on a service or its interface, and *A4* on the operations of that interface.

Annotation A5: Nonfunctional annotations of WSDL Service. Nonfunctional descriptions apply to a concrete instance of a Web service, that is, a WSDL Service. A reference annotation can point from a service component to a nonfunctional property. Listing 5.4 shows an example of annotation *A5*, attaching pricing information to a Web service (line 3).

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

Listing 5.4: Example of annotation A5

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

6 Minimal RDF Representation of WSMO-Lite Service Descriptions

In this chapter, we define a minimal representation of the semantic information present in WSMO-Lite descriptions that is sufficient for all semantic automation beside Web service invocation. Such a representation can serve as the basis of an efficient index of the semantic descriptions of available services in a service description registry.

The model of our minimal RDF representation is illustrated in Figure 6.1. It only captures terms for the Web service, its operations, and their input, output and fault messages (faults are not shown in the figure for brevity). All these terms are necessary because WSMO-Lite uses them to represent service semantics:

- **Service:** functional semantics are expressed with *A3* annotations on the Web service or on its interface; the distinction between a WSDL service and its interface is irrelevant when considering the functional and nonfunctional semantics of a Web service. Nonfunctional semantics are captured with *A5* annotations on a WSDL service.
- **Operations:** behavioral semantics are described through *A4* annotations of service operations.
- **Messages:** the service's information model is referenced through *A1* annotations on XML Schema components; for the purpose of semantic automation, all these annotations can be folded together for each message that uses those schema components. Lifting and lowering transformations are also attached to those XML Schema components and can be folded onto the appropriate operation messages (including fault messages).

This minimal service model is expressed in RDF using terms defined in Listing 6.1. The RDF vocabulary includes three classes (Service, Operation and Message) and five properties (hasOperation, hasInputMessage, hasInputFault, hasOutputMessage, hasOutputFault). Further, the listing includes the three RDF properties defined by SAWSDL, which are used to express the WSMO-Lite annotations in this minimal RDF representation.

The listing also defines an additional property usesOntology that is employed for identifying ontologies that may be relevant when reasoning about the semantics of a given service. This is particularly necessary because SAWSDL allows the definitions of the semantic concepts used by the annotations to be included in the WSDL document, for instance, a service

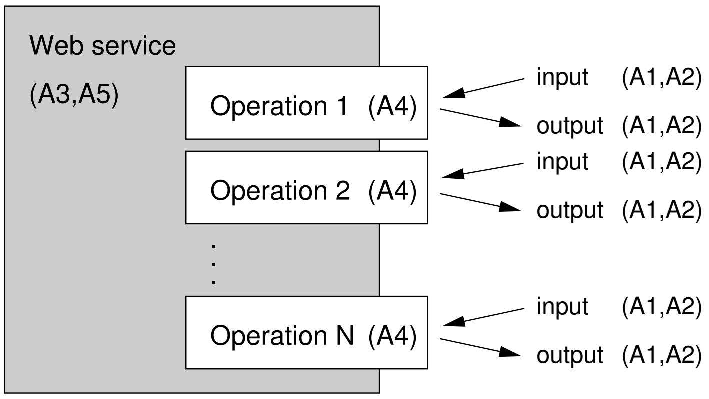

Figure 6.1: Minimal Web Service Model with WSMO-Lite Annotations

```
# namespace declarations
@prefix rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf−schema#> .
@prefix sawsdl: <http://www.w3.org/ns/sawsdl#> .
@prefix wsl: <http://www.wsmo.org/ns/wsmo−lite#> .

# service model classes and properties
wsl:Service a rdfs:Class .
wsl:hasOperation a rdf:Property ;
    rdfs:domain wsl:Service ;
    rdfs:range wsl:Operation .
wsl:Operation a rdfs:Class .
wsl:hasInputMessage a rdf:Property ;
    rdfs:domain wsl:Operation ;
    rdfs:range wsl:Message .
wsl:hasOutputMessage a rdf:Property ;
    rdfs:domain wsl:Operation ;
    rdfs:range wsl:Message .
wsl:hasInputFault a rdf:Property ;
    rdfs:domain wsl:Operation ;
    rdfs:range wsl:Message .
wsl:hasOutputFault a rdf:Property ;
    rdfs:domain wsl:Operation ;
    rdfs:range wsl:Message .
wsl:Message a rdfs:Class .

# SAWSDL properties (included here for completeness)
sawsdl:modelReference a rdf:Property .
sawsdl:liftingSchemaMapping a rdf:Property .
sawsdl:loweringSchemaMapping a rdf:Property .

# property for identifying potentially relevant ontologies
wsl:usesOntology a rdfs:Property ;
    rdfs:domain wsl:Service ;
    rdfs:subPropertyOf rdfs:seeAlso .
```

Listing 6.1: Service Model, captured in Notation 3

```
@prefix ex: <http://example.com/svc.wsdl#>
@prefix rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf−schema#> .
@prefix sawsdl: <http://www.w3.org/ns/sawsdl#> .
@prefix wsl: <http://www.wsmo.org/ns/wsmo−lite#> .

ex:ExampleCommLtd a wsl:Service ;
    rdfs:isDefinedBy <http://example.org/svc.wsdl> ;
    sawsdl:modelReference <http://example.org/onto#VideoOnDemandPrice> ;
    sawsdl:modelReference <http://example.org/onto#VideoSubscriptionService> ;
    sawsdl:modelReference <http://example.org/onto#VideoOnDemandSubscriptionPrecondition> ;
    sawsdl:modelReference <http://example.org/onto#VideoOnDemandSubscriptionEffect> ;
    wsl:hasOperation ex:CheckNetworkConnection .

ex:CheckNetworkConnection a wsl:Operation ;
    wsl:hasInputMessage ex:CheckNetworkConnectionRequest .

ex:CheckNetworkConnectionRequest a wsl:Message ;
    sawsdl:modelReference <http://example.org/onto#NetworkConnection> ;
    sawsdl:loweringSchemaMapping <http://example.org/NetCn.xslt> .
```

Listing 6.2: Example service data in the minimal RDF representation

may specify its precondition and effect in WSML and include their definitions in the XML syntax of WSML in an extension element on the WSDL description root element. Commonly, ontology element URIs are resolvable to their defining ontologies, but this is unlikely to be the case for ontologies embedded in WSDL descriptions. Therefore, any ontology fragment embedded in a WSMO-Lite document will be referenced from the service in our RDF representation with the usesOntology property, so that they can be included in reasoning.

Finally, automating Web service invocation needs access to the original WSDL document, in particular the endpoint and binding information. This information is deliberately omitted in our minimal RDF WSMO-Lite representation; instead, the service in the RDF representation points to the original WSDL document using the RDFS property isDefinedBy [11].

Listing 6.2 shows what the RDF representation of the WSDL data included in the examples in Listings 5.2, 5.3 and 5.4.

7 Viewing WSMO-Lite as WSML

In this chapter, we show how a WSMO-Lite description (or more precisely, a WSDL description with SAWSDL annotations pointing to WSMO-Lite-compliant semantics) can be viewed as a WSML [18] description.

Table 7.1 shows the five kinds of annotations of WSMO-Lite and their WSML counterparts. Information model annotations of the messages (*A1*) translate into choreography state signature when a choreography is generated from functional annotations of operations (*A4*), as described in [19]. Lifting and lowering annotations (*A2*) are out of scope of WSMO, they are treated as grounding details. Precondition and effect annotations *A3* become preconditions and effects of a WSMO capability description (which has no assumptions or postconditions). Functionality categorization annotations *A3* must be represented as nonfunctional property of the capability of the Web service in WSMO since there is no native support for functionality categories in WSMO. And finally, nonfunctional annotations *A5* are represented as nonfunctional properties of the Web service in WSMO, under the general WSMO-Lite NFP name wsl#nonFunctionalParameter[1]

This mapping can be used to load WSMO-Lite descriptions into WSML tooling, such as WSMX, for reuse of components such as service discovery and ranking. However, WSMO does not support the coarse-grained service functionality classifications used by WSMO-Lite, therefore, existing WSMX components (at the time of this writing) will not be able to use the functionality classifications, which are represented as capability NFPs in the WSML form.

WSMO-Lite annotation	**WSML counterpart**
generated for every WSDL service	webservice *svcID* *// generated service identifier* nonFunctionalProperty wsml#endpointDescription hasValue *wsdl-service-uri*
A1	*Information model annotations can be reflected in a choreography state signature, see mapping of A4.*
A2	*Lifting/lowering annotations are out of scope for WSMO.*
A3: condition *P*, effect *E*	webservice *svcID* capability *capID* *// generated capability identifier* precondition *P* effect *E*
A3: functionality category *C*	*Becomes a nonfunctional property:* webservice *svcID* capability *capID* nonFunctionalProperty wsl#functionalityCategory hasValue *C*
A4: operation capabilities	*Service choreography, see mapping algorithm in [19].*
A5: nonfunctional parameter *N*	webservice *svcID* nonFunctionalProperty wsl#nonFunctionalParameter hasValue *N*

Table 7.1: Mapping of WSMO-Lite into WSML

[1]Nonfunctional properties are represented in WSMO with a tuple (property name, value), whereas in WSMO-Lite they are ontology individuals. Therefore we introduce the property name wsl#nonFunctionalParameter for WSMO-Lite NFPs in WSML.

8 Related work

The major stream of related work is in the frameworks for Semantic Web Services (SWS), including WSMO [12], Semantic Markup for Web Services (OWL-S [17]) 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 interlinked ontologies: Service Profile (for the functional and nonfunctional 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, 5]. 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-S[1] 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. The core parts of WSDL-S were taken as the basis for SAWSDL; WSDL-S also included WSDL extensions for attaching preconditions, effects and categories; however, they were out of scope for SAWSDL as they can be moved to ontologies and attached through model references, as we do in WSMO-Lite.

In addition, there is a major orthogonal work to WSMO-Lite called hRESTS and MicroWSMO (CMS WG deliverable D12, also [3]), aiming to enrich the informal descriptions of RESTful services, usually available in HTML, with microformat or RDFa [8] annotations. Effectively, hRESTS forms an analogue of WSDL for RESTful services, and MicroWSMO is analogous to SAWSDL. The WSMO-Lite service semantics ontology is directly applicable in MicroWSMO and hRESTS annotations.

[1] `http://lsdis.cs.uga.edu/projects/meteor-s/`

9 Conclusions and Future Work

In this deliverable, 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 [4]: “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 and on implementing SWS automation with WSMO-Lite descriptions. We also plan 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.

ACKNOWLEDGEMENTS

The authors would like to thank to all the members of the Conceptual Models for Services working group[1] for their advice and input to this document.

[1] `http://cms-wg.sti2.org/operation/members/`

REFERENCES

[1] Rama Akkiraju, Joel Farrell, John Miller, Meenakshi Nagarajan, Marc-Thomas Schmidt, Amit Sheth, and Kunal Verma. Web Service Semantics – WSDL-S, available at http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/. Technical report, LSDIS Lab, 2005.

[2] Ian Horrocks. OWL: A Description Logic Based Ontology Language. In *Proc. of the 21st International Conference on Logic Programming*, pages 1–4. Springer Verlag, 2005.

[3] Jacek Kopecký, Karthik Gomadam, and Tomas Vitvar. hRESTS: an HTML Microformat for Describing RESTful Web Services. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence (WI-08)*, Sydney, Australia, 2008. To appear.

[4] David Martin and John Domingue. Semantic web services: Past, present and possible futures (systems trends and controversies). *IEEE Intelligent Systems*, 22(6), 2007.

[5] 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.

[6] Anthony Nadalin, Chris Kaler, Phillip Hallam-Baker, and Ronald Monzillo. Web Services Security: SOAP Message Security 1.0 (WS-Security 2004). OASIS Standard, March 2004. Available at http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.

[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 *LNCS*, pages 416–421. Springer, 2007.

[8] RDFa in XHTML: Syntax and Processing. Recommendation, W3C, October 2008. Available at http://www.w3.org/TR/rdfa-syntax/.

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

[10] Resource Description Framework (RDF): Concepts and Abstract Syntax. Recommendation, W3C, February 2004. Available at http://www.w3.org/TR/rdf-concepts/.

[11] RDF Vocabulary Description Language 1.0: RDF Schema. Recommendation, W3C, February 2004. Available at http://www.w3.org/TR/rdf-schema/.

[12] Dumitru Roman, Uwe Keller, Holger Lausen, Jos de Bruijn, Ruben Lara, Michael Stollberg, Axel Polleres, Cristina Feier, Cristoph Bussler, and Dieter Fensel. Web Service Modeling Ontology. *Applied Ontology*, 1(1):77–106, 2005.

[13] Dumitru Roman, James Scicluna, Dieter Fensel, Axel Polleres, and Jos de Bruijn. Ontology-based Choreography of WSMO Services. Wsmo d14 final draft v0.3, DERI, 2006. Available at: http://www.wsmo.org/TR/d14/v0.3/.

[14] Semantic Annotations for WSDL and XML Schema. Recommendation, W3C, August 2007. Available at http://www.w3.org/TR/sawsdl/.

[15] Amit P. Sheth. Semantic Web Process Lifecycle: Role of Semantics in Annotation, Discovery, Composition and Orchestration. Invited Talk, Workshop on E-Services and the Semantic Web, at WWW 2003. Available at `http://lsdis.cs.uga.edu/lib/presentations/WWW2003-ESSW-invitedTalk-Sheth.pdf`.

[16] R. Studer, S. Grimm, and A. Abecker. *Semantic Web Services: Concepts, Technologies, and Applications*. Springer-Verlag New York, Inc. Secaucus, NJ, USA, 2007.

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

[18] Ioan Toma and Nathalie Steinmetz (eds.). The Web Service Modeling Language WSML. Available at `http://www.wsmo.org/TR/d16/d16.1/v0.3/`, 2008.

[19] Tomas Vitvar, Jacek Kopecký, Jana Viskova, and Dieter Fensel. WSMO-Lite Annotations for Web Services. In *Proceedings of 5th European Semantic Web Conference (ESWC)*, 2008.

[20] Web Services Agreement Specification (WS-Agreement). Proposed Recommendation, Global Grid Forum, September 2005. Available at http://www.ggf.org/Public_Comment_Docs/Documents/Oct-2005/WS-AgreementSpecificationDraft050920.pdf.

[21] Web Services Business Process Execution Language Version 2.0. OASIS Standard, April 2007. Available at `http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html`.

[22] 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/`.

[23] Web Services Policy 1.5 – Framework. Recommendation, W3C, September 2007. Available at http://www.w3.org/TR/ws-policy/.

[24] WS-Reliability 1.1. OASIS Standard, 2004. Available at `http://docs.oasis-open.org/wsrm/ws-reliability/v1.1/wsrm-ws_reliability-1.1-spec-os.pdf`.