SemEUsE

SEMantiquE pour bUS de sErvice

2007 TLOG 018

Document name:	Functional properties, user preferences and context information semantic description
Document version:	0.2
Task code:	T2.1
Deliverable code:	T2.1D1
WP Leader (organisation):	INT
Deliverable Leader (organisation):	INT
Authors (organisations):	INT, INSA, TCF
Date of first version:	27/07/2008

Change control

Changes	Author / Entity	Code of version
Creation of the document 15/05/08	INT	V0
Revision of the document after meeting 19/05/08	INT	V0.1
Revision and extension of the document	INT	V0.2

Table of Contents

1. Introduction

Web services are modular, loosely-coupled and self descriptive applications, providing a simple model of programming and deployment of applications. They are running through the Web and based on standards such as SOAP for message transport, WSDL for service description, and UDDI for service advertisement and discovery. Nevertheless, the lack of semantics in WSDL (W3C, 2001; Chinnici et al., 2007) prevents automatic discovery and hence automatic invocation and composition. To deal with these issues several approaches were developed that use semantic models (ontologies,...) for description of semantic Web services. We can cite among others, OWL-S (W3C, 2004), SAWSDL (Farrell & Lausen, 2007) and WSMO (ESSI WSMO working group, 2004).

WSMO propose high-level objectives and approaches similar to those of OWL-S but WSMO focuses on goals mediation and choreography. OWL-S focuses on process model. In SAWSDL there is no explicit mention of precondition and effects that one can find in WSMO and OWL-S. In addition SAWSDL is not dedicated to describe Web service behaviour which is essential for service invocation and composition. Nevertheless, SAWSDL is an approach independent of the used semantic representation language thanks to the separation of semantic annotation mechanism from the representation of the semantic descriptions. This gives flexibility to developer community to select their favourite semantic representation language, to reuse semantic domain models and annotate descriptions using multiple ontologies.

In our work, we are interested in description of semantic Web services taking into account existing approaches.

It is obvious that our proposed semantic description of Web service functional properties should be based on the de-facto standard to describe Web services, namely Web Services Description Language (WSDL[1]). Ideally, the proposed specification would require no other changes to existing WSDL or XML Schema documents (Fallside & Walmsley, 2004), or the way in which they had been used previously. Specially, the developed tools for parsing WSDL documents or for invoking Web services based on their WSDL descriptions still in this case usable. To meet this target, we propose YASA4WSDL an extension of SAWSDL that use two types of ontologies. The first one, called Technical Ontology Type, concerns ontologies that describe service concepts (e.g. interface, input, output...) and ontologies that describe non functional concepts of services (e.g. QoS, security, context awareness attribute types...). The second ontology type, called Domain Ontology Type, concerns ontologies that define the semantics of the service business domain (e.g. tourism, health, trade...).

This document is organized as follows. Section 2 presents WSDL and SAWSDL languages which our proposed description is based on. In addition, this section presents an overview of OWL-S and WSMO approaches for semantic service description that inspired our proposition. Section 3 presents requirements we have identified for functional properties description. Section 4 presents our semantic description that fulfills the identified requirements.

[1] WSDL 2.0 Home Page. http://www.w3.org/TR/wsdl20

2. Related Work

We present in this section the most related work to our contribution: SAWSDL, OWL-S, WSMO, WSMO-Lite and CC/PP. Our contribution according the works presented here is explained in Section 4.

2.1. WSDL

Web Services Description Language provides a model and an XML format for describing Web services. It separates the description of the abstract functionality offered by a service from concrete details of a service description such as ”how” and ”where” that functionality is offered.

- The abstract part of the description defines data types, operations and interfaces. An operation associates one or more inputs and outputs, the Web service sends and receives, with a message exchange pattern (MEP) that defines the sequence and cardinality of messages. An interface groups operations in a transport and wires independent manner.
- The concrete part of the description defines bindings, service endpoint and a service. Bindings specify the transport and wire format for interfaces. A service endpoint associates network address with a binding. Finally, a service groups the endpoints that implement a common interface.

WSDL 2.0 extension takes place by adding to WSDL document XML elements and attributes from XML namespaces (non-WSDL). XML namespaces are used to report on the structure and semantics of the new extensions for example new elements/attributes’ names and lengths.

2.2. SAWSDL

In (Chinnici et al., 2007), SAWSDL suggests how to add semantic annotations to various parts of a WSDL document like input and output message structures, interfaces, and operations. This extension is inline with WSDL extensibility framework. SAWSDL defines a new namespace called *sawsdl* and adds an extension attribute called *modelReference* so that relationships between WSDL components and concepts in another semantic model (e.g. ontology) are handled.

The *modelReference* annotations on *xs:element*, *xs:complexType*, *xs:simpleType* and *xs:attribute* define the semantics of the input or output data of WSDL operations. A *modelReference* on a WSDL operation or fault gives semantic information about that operation, while a *modelReference* on a WSDL interface provides a classification or other semantic descriptions of the interface (e.g. non-functional properties).

The modelReference attribute may contain a list of references, but actually, one can not know the nature or concept of each reference, so one can not link the semantic information to a particular item in the vocabulary of service (profile, process, ...) or to a feature of the service (effect of operation, pre-condition

2.3. OWL-S

OWL-S approach proposes an ontology of services motivated by the need to provide three essential types of knowledge about a Web service each of them characterized by the question it answers:

- What does the service provide for prospective clients? The answer to this question is given in the "profile" which is used to advertise the service.
- How is it used? The answer to this question is given in the "process model."
- How does one interact with it? The answer to this question is given in the "grounding." A grounding provides the needed details about transport protocols.

In OWL-S, functional properties of Web services are mainly described by Service Profile and Process.

A Service Profile is used for populating service registries, automation of service discovery, and selection. As a consequence one can derive from a service profile service requests. The service profile elements include:

- Preconditions: Set of conditions that should hold prior to service invocation
- Inputs: Set of necessary inputs that the requester should provide to invoke the service
- Outputs: Set of outputs that the requester should expect after interaction with the service provider is completed

- Result: Effects and condition for delivery. Effects are a set of statements that should hold true if the service is invoked successfully.
- Service category: categories of services on the basis of some classification that may be outside OWL-S and possibly outside OWL.

A Service Process is used for service invocation, planning/composition. Its elements include

- Inputs, outputs, preconditions and effects
- Behaviour of the service (data and control flow)

2.4. WSMO

In (ESSI WSMO working group, 2004), WSMO is a conceptual model for four top level elements as the main concepts which have to be described in order to describe Semantic Web services: ontologies, services, mediators and goals.

Descriptions of a WSMO service include˜ non functional properties, a provided interface and a provided capability. Descriptions of a WSMO goal include˜ non functional properties, a requested interface and a requested capability.

A WSMO interface describes messages sent to/by a WSMO service and the visible behavior of that service.

A WSMO capability includes:

- non functional properties,
- pre-conditions : what a web service expects in order to be able to provide its service. They define conditions over the input,
- assumptions : conditions on the state of the world that has to hold before the Web Service can be executed and work correctly, but not necessarily checked/checkable,
- post-conditions : describe the result of the Web Service in relation to the input, and conditions on it,
- effects : conditions on the state of the world that hold after execution of the Web Service (i.e. changes in the state of the world).

2.5. WSMO Lite

Another effort in enhancing Web service description with semantics is WSMO-Lite (Vitvar et al., 2007). Contrary to WSMO and OWL-S, it adopts a bottom-up approach to modelling services and is built on SAWSDL. WSMO-Lite identifies vocabulary for semantic description and proposes a bridge between WSDL, SAWSDL and domain-specific ontologies. But WSMO-Lite does not prescribe concrete language for functional semantics and as effects and conditions are language-dependent, it cannot specify semantics for them (Vitvar et al., 2007).

2.6. Other contributions on description of semantic Web services

In (Martin et al., 2007b), the authors propose to use OWL-S constructs (service profile and process model) as references of SAWSDL annotations. The idea is not to continue employing the OWL-S grounding and to adopt a SAWSDL-based perspective. This approach starts from the assumption that atomic processes in OWL-S correspond to SAWSDL's operations. Then it propose to add a *modelReference* attribute with message elements as it is not defined by SAWSDL, and finally the authors propose to refer a message's *modelReference* to an OWL class. However, in this paper, there are many issues needing clear decision and more precision, having to do with the mapping of the inputs, outputs, operations, interfaces, and faults. It is related with difficulties in mapping between Message Exchange Pattern in SAWSDL and inputs/outputs in atomic processes on one hand and the fact that OWL-S does not have constructs that provide direct correspondence to "interface" elements in SAWSDL on the other hand.

In (Li &Wang, 2006), the Composite Capabilities/Preference Profiles (CC/PP) standard vocabulary (Klyne et al., 2004) is extended and semantic information is added to the description. The W3C proposed standard has a description profile constructed as a two-level hierarchy: a set of components having each one of them at least one or more attributes. So in order to describe more service information, new CC/PP components and new attribute vocabularies (type of service, quality of service, location, and additional information, etc) are introduced in (Li & Wang, 2006). Actually capability is a useful concept but it does not satisfy all the user needs about service description such as process aspect of services or operation effects.

In (Klein et al., 2005; Küster & Konig-Ries, 2007), DIANE Service Description (DSD) and DIANE Elements (DE) propose object-oriented service description in order to put into practice additional requirements that are not fulfilled by semantic service description such us WSMO and OWL-S. In (Klan, 2006; Küster et al., 2007), the authors describe how DSD is enhancing semantic description by using a ontology to express concepts like precondition and effect.

2.7. Discussion

WSMO proposes high-level objectives and approaches similar to those of OWL-S but WSMO focuses on goals, mediation, and on choreography but not on process model.

In SAWSDL there is no explicit mention of precondition and effects that one can find in WSMO and OWL-S. In addition SAWSDL is not dedicated to describe Web service behavior which is essential for service invocation and composition.

SAWSDL is an approach independent of the used semantic representation language thanks to the separation of semantic annotation mechanism from the representation of the semantic descriptions. This gives flexibility to developer community to select their favorite semantic representation language, to reuse semantic domain models and annotate descriptions using multiple ontologies.

The WSDL extensibility mechanism and SAWSDL-based semantic annotations need to be faced to our requirements for possible extensions in order to address limitations we cited above, and take advantage of SAWSDL flexibility and incremental aspect. Section 3 and Section 4 are respectively dedicated to the identification of functional description requirements and propose a semantic description of Web services functional properties within the SemEUsE project.

3. Functional description requirements

Semantic Web services use domain ontologies to provide their description in terms of semantic concepts. A concept is used to denote service description parts in the service domain ontology. Semantic-based description languages such us OWL-S and WSMO are closed approaches: they only handle respectively OWL and WSML as type of ontologies (Ould Ahmed M'Bareck & Tata, 2008). Moreover, they specify a definite but limited set of concepts which is not easy to extend. WSMO contains some concepts that do not appear in OWL-S ontology and vice versa. In SAWSDL there is no explicit mention of *precondition* and *effects* that one can find in WSMO and OWL-S. In addition SAWSDL is not dedicated to describe Web service behavior which is essential for service invocation and composition.

Semantic annotation in SAWSDL use an extended attribute called *modelReference* so that relationships between WSDL components and concepts in another semantic model (e.g. ontology) are handled. Therefore, SAWSDL is an approach independent of the semantic representation language thanks to the separation of semantic annotation mechanism from the representation of the semantic descriptions. This gives flexibility to the developers' community to select their favorite semantic representation language, to reuse semantic domain models and annotate descriptions using multiple ontologies.

In order to enhance service discovery, composition and invocation, we have identified requirements for the description of semantic Web services: differentiation for semantic annotation of WSDL elements, behavior description and supporting several implementations/interfaces for a given Web service functionality.

3.1. Differentiation of semantic annotation of WSDL elements

In order to enhance service discovery, composition and invocation, we have identified requirements for the description of semantic Web services. The SAWSDL specification states that a *modelReference may be used with every element within WSDL and XML schema. However, SAWSDL defines its meaning only for interface, operation, fault, element, complexType, simpleType and attribute* (Farrell & Lausen, 2007*). 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* (Martin et al., 2007a). 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*, and similarly for operations. Consequently, when an operation is annotated using several semantic concepts in a ontology, one is not able to differentiate these concepts: which one annotates category, which one annotate the behavior, which one annotate the QoS? And so on. Similar observation can be made for any semantic annotation of any WSDL element. Therefore, we are convinced that it is necessary to be able to differentiate the semantic description of all services elements. Differentiation of semantic annotation of WSDL elements can be used to enhance Web service discovery. Instead of using *modelReference* to associate one or more semantic properties to a WSDL element, we should use means to differentiate each semantic property that can be associated to a WSDL element. Indeed, one can consider discovering Web services using one specific semantic property such as Web service effects or can consider composing/invoking Web services using one specific semantic property such as Web service behavior.

3.2. Behavior model description

Consider a Web service along with a set of operations. One can define for that same set of operation more than control flow-based relationship. Service discovery and composition should take into account, in this case, the behavior relation between operations. To address this issue, a solution could be realized using a reference to an external behavior model (kind of OWL-S process model or WSMO interface) that provides a description of how to interact with the service at an abstract level. A behavior model can be used to enhance semantic-based Web service discovery and composition.

3.3. Supporting several implementations and/or interfaces for one functionality

According to WSDL 2.0's specification, a *wsdl:service* element specifies a single *wsdl:interface* that the Web service will support, and a list of *wsdl:endpoint* locations where that Web service can be accessed. Each endpoint must also reference a previously defined binding to indicate what protocols and transmission formats are to be used at that endpoint. A service is only permitted to have one interface. As for us, several independent implementations realizing different capabilities with their respective interfaces can be used to satisfy a unique functionality of a Web service in different ways in order to meet several QoS values, validity contexts, user preferences, etc. Building on top of WSDL specification (2.0), we propose new means to describe first, several interfaces a Web service supports, and second, a list of endpoint locations, one per interface binding (implementation), that give access to the interfaces of this Web service.

4. Semantic annotation for WSDL

In this section we present the identified semantic functional properties we aim at describing within the SemEUsE project. First of all, we present YASA4WSDL, the proposed extension for SAWSDL. After that, we present an example to illustrate the contribution. Finally we explain the main contribution of YASA4WSDL according to the state of the art.

4.1. YASA4WSDL presentation

The main idea is to extend SAWSDL for enhancing expressiveness of service description. In SAWSDL for a given WSDL element one can use many references to concepts in a domain ontology but there is no specification of the semantic information nature: is it a precondition, an effect, a result? etc. That is why we propose, in our description, a new attribute called *serviceConcept* to give references to the technical concepts corresponding in the same order, to the domain concepts listed in the original SAWSDL "*modelReference*" attribute. Indeed our approach for semantic Web service description is based on the use of two types of ontologies. The first one, called Technical Ontology, contains several concepts defining semantics of services' concepts and concepts describing their non functional properties (QoS, security, context…). The second type of ontologies, called Domain Ontology, contains the semantics of the service domain concepts (e.g. tourism, health, trade...).

A service described in YASA4WSDL can define for each WSDL element two attributes providing semantic description. The first attribute, called *serviceConcept*, contains a set of URI referencing the corresponding concepts in one or several Technical Ontologies. The second attribute contains a set of URI corresponding to the first list and which define the semantics in one or several Domain Ontologies. Let's consider the example presented in Figure 1. There are two ontologies: the first is Technical (Service Ontology) that describes the semantics of some service concepts: *precondition* and *effect*. The second ontology is a Domain ontology (called *TransportOntology*) that describes some concepts in the travel domain: *validFlightInfo* and *reservationInfo*. The example presented the semantic annotation of an operation named *reserveFlight*. The *modelReference* attribute references two concepts: *validFlightInfo* and *reservationInfo.* The importance of the extended attribute *serviceConcept* is to distinguish the role played by: *validFlightInfo* and *reservationInfo* referenced by *modelReference* attribute. The *serviceConcept* references two service concepts: *precondition* and *effect* that correspond to the two domain concepts *validFlightInfo* and *reservationInfo.* The order is important here. It associates the first technical concept with the first domain

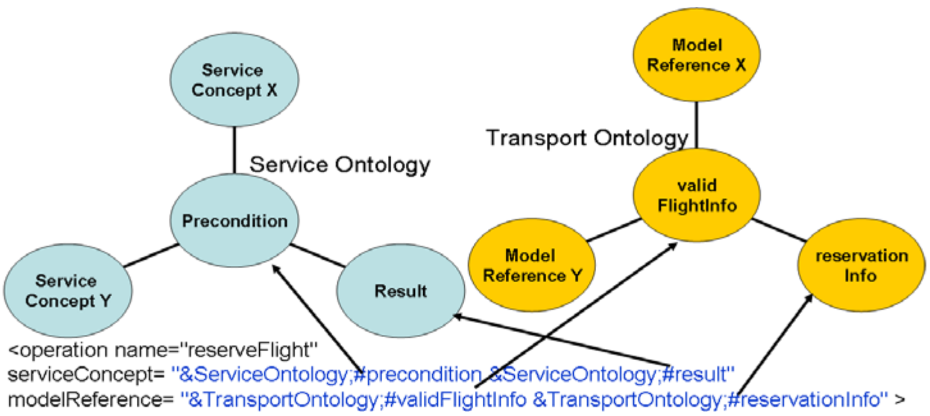

Figure 1. Service Ontology and Domain Ontology

Another advantage is provided by this new approach: in YASA4WSDL, we can extend the Service Ontology by new concepts (new description element, more precise concepts,...) and there is no impact on the annotation system. There will be no limited service ontology. This makes the system of annotation independent of the used semantic business domain representation language and ontology and the used semantic service representation language and ontology. This gives flexibility to the developers' community to select their favourite semantic representation language and their technical ontology, to reuse semantic domain models and annotate descriptions using multiple ontologies.

4.2. Example

The example given below models a Web Service for trip reservation. It provides operation for flight, hotel, and/or car reservation. The service provides flight reservations based on the specification of a flight request. If the desired flight is available, an itinerary and reservation number will be returned. The similar things are offered by the service concerning hotel and car reservation.

As in SAWSDL, a description contains four principal parts: types (lines 2-8), interface(s) (lines 9-20), binding(s) (lines 21-26), and service (lines 27-29). Types component defines *the content model of an element information items such as that defined by an XML Schema global type definition*. "Interface" component is a set of operations and it describes sequences of messages that a service sends and/or receives. "Operation" component is a sequence of input and output messages (lines 15-16). "Binding" component *describes a concrete message format and transmission protocol which may be used to define an endpoint*, it defines the implementation details necessary to access the service. "Service" component describes *a set of endpoints at which a particular deployed implementation of the service is provided* (line 28). "Endpoint" component is place at which the service is provided (Farrell & Lausen, 2007).

The semantic annotation of the interface called TripReservationInterface (line 9) is ensured by two attributes: *serviceConcept* (line 10) and *modelReference* (line11). They indicate respectively that the interface represents a set of reservation operations in the context of a transport domain ontology. In the operation called *reserveFlight* (line 12), her *serviceConcept* attribute (line 13) provides a list of service concepts to which corresponds respectively, in the same order, a list of model references from the transport domain ontology provided by the *modelReference* attribute (line 14). The operation has as *precondition* associated to the concept of *validFlightInfo* (Flight information have to be valid before reserving flight). The operation has as result the concept of *reservationInfo* (Information about reservation is returned after the effective reservation).

```
1  <description>
2      <types>
3      <schema targetNamespace="http://cyl.com/reservation#" elementFormDefault="qualified">
4              <element name="reserveFlightRequest" type="reservation">...</element>
5              <element name="reserveFlightResponse" type="confirmation">...</element>
6              ...
7      </schema>
8      </types>
9      <interface name="TripReservationInterface"
10                     serviceConcept="&ServiceOntology;#interface"
11                     modelReference="&TransportOntology;#reservation">
12             <operation name="reserveFlight" pattern="http://www.w3.org/ns/wsdl/in-out"
13                             serviceConcept="&ServiceOntology;#precondition &ServiceOntology;#result"
14                             modelReference="&TransportOntology;#validFlightInfo &TransportOntology;#reservationInfo">
15                     <input element="reserveFlightRequest"/>
16                     <output element="reserveFlightResponse"/>
17             </operation>
18             <operation name="reserveHotel"...>
19             <operation name="reserveCar"...>
20     </interface>
21     <binding name="reservationSOAPBinding"     interface="TripReservation" type="http://www.w3.org/ns/wsdl/soap"...>
23             <operation ref="reserveFlight"...>
24             <operation ref="reserveHotel"...>
25             <operation ref="reserveCar"...>
26     </binding>
27     <service name="reservationService" interface="TripReservationInterface"...>
28             <endpoint name="reservationEndpoint" binding="reservationSOAPBinding" address="http://cyl.com/reservation"/>
29     </service>
30 </description>
```

4.3. YASA4WSDL contribution

In this subsection we explain the contribution of YASA4WSDL according the sate of the art presented in Section 2 namely OWL-S, WSMO, and WSMO-Lite. Compared to WSMO and OWL-S, YASA4WSDL offers multiple advantages. First, users can describe, in an upwardly compatible way, both the semantics and the syntax of functional and non functional properties in WSDL, a language that the developers' community is familiar with. Second, by externalizing the semantic of business domain models and the technical concepts (service ant its QoS, security, context…), one can allow Web service developers to annotate their Web services with their choice of ontology language (such as UML or OWL) unlike in OWL-S or WSMO. In addition, OWL-S and WSMO define their own Service ontologies unlike in YASA4WSDL, which can integrated any technical ontology (Context, QoS, security or service ontology). In addition (and regarding to the work presented in (Martin et al., 2007b)), YASA4WSDL allows the developer community to specify the correspondence between service concepts (interfaces, operation, fault…) and business concepts. This explicit correspondence can be used to bridge YASA4WSDL and OWL-S, WSMO, and many other description languages for semantic Web services. According to DIANE approach (Klein et al., 2005; Küster & Konig-Ries, 2007), our approach proposes almost the same contributions but it focuses on up to date standards (SAWSDL) and common tools (WSDL-oriented tools).

4.4. Constraints on technical ontologies

References of concepts in service ontologies (type of technical ontologies) should respect a main constraint that we present here. Indeed, for a given WSDL element (service, interface, operation…) one can not reference any concept in a service ontology. For example, a behavior concept (that can be a process in an OWL service ontology) can be only associated to interface and operation elements. A behavior concept can not be associated to an input. Consequently, a service ontology should be an extension of the metamodel of WSDL (concepts of Web service like service, interface, operation,…) with additional concepts that characterize (king of *hasproperty* relation) the WSDL concepts. With this condition, the YASA4WSDL *serviceConcept* attribute of a WSDL element *E* should only references concepts that are related to the concept *E* in the service ontology. In the SemEUsE project we should develop a service ontology that is an extension of the WSDL metamodel and contains additional concepts inspired by OWL-S and WSMO ontologies.

We list in the following the functional properties we have identified. For each property we explain the WSDL elements it can characterize. Some of these properties are inspired by OWL-S and some others are inspired by WSMO.

4.4.1. Behaviour model

This property is inspired by the OWL-S process model. It describes the behavior (data and control flow) associated to a given Web service interface. Consequently, this property can be only associated to interface element.

4.4.2. Precondition

This property is inspired by the WSMO precondition property. It describes what a web service operation or operations of an interface expect, in order to be able to provide its/their functions. Preconditions define conditions over a given input declaration. This property can be associated to an input declaration of operation element. It can be associated to an operation (respectively interface) elements when conditions are associated to all input declarations of operation (respectively interface) elements.

4.4.3. Post-condition

This property is inspired by the WSMO post-condition property. It describes the result of the Web Service operation in relation to its inputs, and preconditions on it. This property can be associated to operation elements. When it is associated to interface elements, preconditions describe conditions on all results of interface operations in relation to all their inputs, and precondition of interface or precondition of all operations of the interface.

4.4.4. Effect

This property is inspired by the WSMO effect property. It can be only associated to operation and interface WSDL elements.

4.4.5. Assumptions

This property is inspired by the WSMO assumption property. It describes conditions on the state of the world that has to hold before the Web Service can be executed and work correctly. Contrary to WSMO approach the assumptions should necessarily be checkable and checked before Web service invocation. This property can be used to support context-awareness. Assumptions can describe contexts in which the service can be invoked. Consequently this property can be only associated to interface and service elements.

To be discussed: authorize or not that a service and its interfaces are annotated by the assumption property. If yes how service and interfaces assumptions are computed? Conjunction or priority to interface (verify the specific before the general).

5. References

[Chinnici et al., 2007]Chinnici, Roberto, Weerawarana, Sanjiva, Moreau, Jean-Jacques, & Ryman, Arthur. 2007 (June). Web services description language (WSDL) version 2.0 part 1: Core language. W3C recommendation. W3C. http://www.w3.org/TR/2007/REC-wsdl20-20070626.

[ESSI WSMO working group, 2004]ESSI WSMO working group. 2004. Web Service Modeling Ontology. http://www.wsmo.org/.

[Fallside & Walmsley, 2004]Fallside, David C., & Walmsley, Priscilla. 2004 (Oct.). XML schema part 0: Primer second edition. W3C recommendation. W3C. http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/.

[Farrell & Lausen, 2007]Farrell, Joel, & Lausen, Holger. 2007 (Aug.). Semantic annotations for WSDL and XML schema. W3C recommendation. W3C. http://www.w3.org/TR/2007/REC-sawsdl-20070828/.

[Klan, 2006]Klan, Friederike. 2006. Context-aware service discovery, selection and usage. Pages 95–99 of: Grundlagen von datenbanken.

[Klein et al., 2005]Klein, Michael, Ries, Birgitta Konig, & Mussig, Michael. 2005. What is needed for semantic service descriptions, a proposal for suitable language constructs. Int. j. web grid serv., 1(3/4), 328–364.

[Klyne et al., 2004]Klyne, Graham, Butler, Mark H., Reynolds, Franklin, Ohto, Hidetaka, Woodrow, Chris, Hjelm, Johan, & Tran, Luu. 2004 (Jan.). Composite capability/preference profiles (CC/pp): Structure and vocabularies 1.0. W3C recommendation. W3C. http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/.

[Küster & Konig-Ries, 2007]Küster, U., & Konig-Ries, B. 2007. Semantic Mediation between Business Partners - A SWS-Challenge Solution Using DIANE Service Descriptions. Web Intelligence and Intelligent Agent Technology Workshops, 2007 IEEE/WIC/ACM International Conferences on, Nov., 139–143.

[Küster et al., 2007]Küster, Ulrich, K¨onig-Ries, Birgitta, Stern, Mirco, & Klein, Michael. 2007. DIANE: an integrated approach to automated service discovery, matchmaking and composition. Pages 1033–1042 of: Www ’07: Proceedings of the 16th international conference on world wide web. New York, NY, USA: ACM.

[Li & Wang, 2006]Li, Hui, & Wang, Haiyang. 2006. A Method of Service Description and Discovery in Pervasive Computing Environments. 1st International Symposium on Pervasive Computing and Applications, Xinjiang, China, 604–607.

[Martin et al., 2007a]Martin, David, Paolucci, Massimo, & Wagner, Matthias. 2007a. Bringing Semantic Annotations toWeb Services: OWL-S from the SAWSDL Perspective. Pages 340–352 of: ISWC/ASWC.

[Martin et al., 2007b]Martin, David, Paolucci, Massimo, & Wagner, Matthias. 2007b. Toward Semantic Annotations of Web Services: OWL-S from the SAWSDL Perspective. In: OWL-S: Experiences and Directions Workshop, the 4th European Semantic Web Conference (ESWC 2007), the Tyrol region of Innsbruck, Austria.

[Ould Ahmed M’Bareck & Tata, 2008]Ould Ahmed M’Bareck, Nomane, & Tata, Samir. 2008. Services Web : revue des approches de description sémantique. Conférence internationale ”Systèmes d’Information et Intelligence Economique”, Hammamet, Tunisie.

[Vitvar et al., 2007]Vitvar, Tomas, Kopeck´y, Jacek, Zaremba, Maciej, & Fensel, Dieter. 2007. WSMO-Lite: Lightweight Semantic Descriptions for Services on the Web. ECOWS ’07. Fifth IEEE European Conference on Web Services, Halle, Germany, Nov., 77–86.

[W3C, 2001]W3C. 2001. Web Services Description Language (WSDL). http://www.w3.org/TR/wsdl/.

[W3C, 2004]W3C. 2004. OWL-S: Semantic Markup for Web Services. http://www.w3.org/Submission/OWL-S/.