

IRS-III Supporting the World of Semantic Web Services

Farshad Hakimpour¹, John Domingue¹, Liliana Cabral¹ and Denilson Sell²

¹Knowledge Media Institute, the Open University, Milton Keynes, UK
f.hakimpour@hakimpour.com, {j.b.domingue, l.s.cabral}@open.ac.uk

²Stela Group and INE, Universidade Federal de Santa Catarina, Brazil
denilson@stela.ufsc.br

Abstract. OWL-S and WSMO are two prominent initiatives that present specifications for Semantic Web Services. IRS-III is the first WSMO compliant implemented infrastructure to support Semantic Web Services. Furthermore, IRS-III provides support for the OWL-S service descriptions by importing the descriptions to IRS-III. This paper describes how the underlying model of IRS-III (WSMO) can support OWL-S descriptions. Through our work we explored different aspects of the two specifications and point out their similarities and differences.

1 Introduction

Semantic Web Services technology lays its foundation on both Semantic Web [1] and Web services. Stack of Web services technologies (i.e. SOAP, WSDL, UDDI, etc.) present specifications that cover details required for an automated interoperation among client agents and services with a minimum interference of a human agent. A Web service may be providing a process (e.g. unit conversion or currency exchange), providing data (e.g. selling statistical or geographic data) or providing a non-digital service (e.g. selling a book or reserving a flight). Important issues in interaction with a Web service include not only the technical interaction issues, but also business negotiation, including finding a suitable service, employing the service and deal with problems and exceptions. Web services demonstrate a high degree of flexibility in the invocation of services and communication with them. Semantic Web offers computer interpretable semantic knowledge that can support a smart selection of services and assist in combining them to build composite services or applications.

We believe Semantic Web Services technology will improve and facilitate *discovery*, *composition* and *interaction* between Web services. Programs on the Web will be able to find each other (other Web services) by matching their requirements with the capabilities of services. Semantic Web Services technology enables Web services to describe their desired requirements and/or provided capabilities. Semantic Web Services facilitates the automatic composition of several Web Services to build a more complex service, while it appears and behaves as a single service to a client agent. This includes aspects related to the provision of a providing to describe a composition. The interaction with Web services not only considers invocation and brokering, but also it would often follow a specific message interchange protocol.

Semantic Web Services technology provides a specification for Web services to describe their interaction pattern, to be used by client agents during the discovery as well as the execution time.

IRS (Internet Reasoning system) is developed to explore the above issues. It complies with Web services technologies and is equipped by a logical reasoning system. The latest version of IRS (IRS-III [2]) is based on WSMO [18] (Web Service Modeling Ontology) specifications and we are expanding it to support OWL-S [10] (Web Ontology Language for Services) specifications. This paper is based on the result of our work to support OWL-S in IRS-III. However, the main purpose of the paper is to explore aspects of the two major specifications in this domain and discuss their similarities and differences –the work is merely based on existing and current releases of the documented specifications.

The rest of this paper is organized as following. In the next three sections we present a brief introduction to each of OWL-S, WSMO and IRS-III. In section 5, we discuss how the semantics of Web services is described by their functional properties in both OWL-S and WSMO. In section 6, we introduce *Mediators*. An understanding of WSMO Mediators is informative before we discuss issues related to Web service composition, in section 7 and introduce current composition models in IRS-III and compare them with OWL-S. Section 8 explains the notion of *Choreography* in WSMO. We discuss *Goals* which has been playing a fundamental role in WSMO and IRS, in section 9 and try to relate it to existing notions in OWL-S. Finally in section 10, we present a summary and conclusion.

In the rest of this paper concepts or relations specific to OWL-S are denoted by Arial narrow and those of WSMO are Capitalized. Also in figures, rounded corner boxes illustrate concepts in OWL-S and boxes show concepts in WSMO. As mentioned above the underlying model for IRS-III is WSMO and in most cases they can be used interchangeably. In case there are extensions to WSMO by IRS-III, those specific features are illustrated by dashed lines. One may also notice that we occasionally refer to IRS (in contrast to IRS-III), where we refer to characteristics related to all former and present versions of IRS.

2 OWL-S

OWL-S [10] is a specification for describing semantics of Web services developed as part of the DAML (DARPA Agent Markup Language, see www.daml.org) program. It was formerly built upon the DAML+OIL, and was known as DAML-S. It is now based on OWL [11] (Web Ontology Language), which is a W3C recommendation for describing ontologies. OWL-S lays its basis on process ontology and benefits from developments in workflow modeling.

OWL-S describes services by three components namely, Service Model, Service Profile and Service Grounding. The significant part of the three is the Service Model that describes its (1) functional parameters, (2) the interaction of a service with its client and its environment, and (3) its execution mechanism in case of a composite service. To that end, OWL-S presents (but is not limited to) ProcessModel. ProcessModel describes a service as a Process (Fig. 1) by its functional parameters: Inputs, Outputs,

Preconditions and Effects (IOPEs). It also specifies the component processes of a composite service and their execution order and binding of the inputs and outputs of component processes.

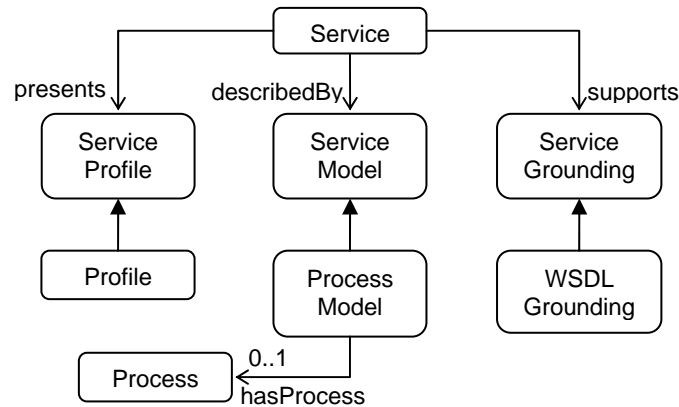


Fig. 1. An overview of main concepts in OWL-S ontology for Web service description.

ServiceProfile foresees information that can be required to search for a service in a service registry. OWL-S Profile consists of information such as, Contact Information of the service providers, ServiceCategory and other non-functional service parameters. Furthermore, a Profile contains a replication of the functional parameters (IOPEs) presented in its ProcessModel. ServiceGrounding specifies the details of accessing and executing a service, such as communication protocol and message format. At present, OWL-S offers specifications for grounding to WSDL descriptions.

3 WSMO

The Web Service Modeling Ontology (WSMO [18], see www.wsmo.org) is a specification for describing the various aspects related to Semantic Web Services. WSMO specification is developing mainly in relation to SDK European project cluster (see www.sdkcluster.org). It is presented in WSML [16] that is a language for formalizing Web service descriptions. WSMO lays its foundation on knowledge representation and logical reasoning and benefits from experiences gained in developing UPML [15] and several case studies (e.g. [19]) in different application domains. The main components of WSMO specifications are

- Goals,
- Web Services,
- Ontologies and
- Mediators.

Goals represent the types of objectives that users would like to achieve via Web services. The WSMO definition of goal describes the state of the desired information space and the desired state of the world after the execution of a given Web service. A goal can import existing concepts and relations defined elsewhere by either extending or simply reusing them as appropriate. A Web service may be selected by discovery process and then executed when achieving a goal is required.

Web services descriptions describe the functional behavior of an actual Web service. WSMO describes Web services by their Capabilities and Interfaces. The interface description contains two closely related notions of Choreography and Orchestration. Choreography describes information required to interact with the Web service. Orchestration can contain information describing a composite Web service.

Ontologies provide the basic glue for semantic interoperability and define the concepts used in the other three components. Finally, WSMO introduces mediators, which specify interoperability mechanisms and that provide the means to link the three components introduced above. The incorporation of mediators in WSMO facilitates the clean separation of different interoperability mechanisms.

4 IRS, the Past and the Present

IRS is a framework and implemented infrastructure for Semantic Web Services that implements the WSMO descriptions. IRS originally was designed as a system to find specific solutions for generic problems in domain of artificial intelligence and now supports major mechanisms needed for Semantic Web Services. We can use IRS for describing and discovering Web Services by their semantics, compose services, invoke them and/or monitor their execution. The IRS consists of three main components, IRS Server, IRS Publisher and IRS Client. IRS Server stores and reasons with the semantic descriptions. It performs reasoning needed for service discovery. The server has also features to monitor the published and running services. IRS Publisher generates wrappers for programs as Web services and links them to the semantic descriptions located on IRS server. IRS client offers an easy to use interface for browsing, editing, describing, publishing, composing and invoking Web services.

IRS service descriptions are in OCML [11]. It is a knowledge representation language by which the ontologies have been formalized. It offers operational modeling capabilities to support quick prototyping of knowledge models. The OCML reasoning system is the basis for the IRS Server.

IRS-II [14] was based on a Task-PSM ontology which followed the UPML framework [15]. As UPML results had major influences on WSMO, IRS-III [2] could support WSMO specification without major modifications to its basic components. IRS-II ontology had two major concepts based on UPML, namely Problem Solving Methods (PSM) and Goal Specification Tasks. The presence of the two concepts helped in providing support for WSMO Goal and Web Service. Separation of the two is one important characteristic of this ontology. A PSM is a description of a method and concerned with the specification of mechanisms and execution. That is where the major similarity between a PSM and the Web Services lays. Goal Specification Tasks in UPML are a general description of a problem and focused on describing the

problem rather than mechanisms. As a result, Goals are suitable for describing a desired service by a user whose concern is not the technical details of the execution of the solution, but rather the required result and the necessary interactions to achieve a solution. IRS-III provides the required support to describe a Goal and supports a Goal-based discovery mechanism.

IRS-III has four main features that distinguish it from other work on Semantic Web Services. Firstly, it supports one-click publishing of ‘standard’ program code. In other words, it automatically transforms programming code into a Web service, by automatically creating an appropriate wrapper. Hence, it is very easy to make existing standalone software available on the net, as Web services. Secondly, by extending the WSMO goal and Web service concepts users of IRS-III directly invoke Web services via goals, supporting capability-driven service invocation. Thirdly, IRS-III is programmable. IRS-III users can substitute their own Semantic Web Services for some of the main IRS-III components. Finally, IRS-III services are Web service compatible –standard Web services can be trivially published through the IRS-III and any IRS-III service automatically appears as a standard Web service to other Web service infrastructures.

5 Web Service Capability

WSMO describes Web Services by two components: *Interface* and *Capability*. Capability describes the functional properties of a service by means of a set of conditions to hold before its invocation and a set of conditions that will hold after its execution. Interface descriptions have two major parts namely *Orchestration* and *Choreography* (Fig. 2). The Web service Interface determines the information needed for executing and interacting with the Web service. For a composite service, Orchestration presents a description of composition of Goals (Sec. 9) or Web services. Choreography describes the pattern of interaction with the Web service and its grounding. In this section we focus on the Capability description. Orchestration and Choreography are discussed further in sections 7 and 8, respectively.

Capability is a key concept for describing semantics of Web services. Its descriptions appear in both WSMO Goal and WSMO Web service specifications [18]. In a Web service Capability determines the service provided and in a Goal it

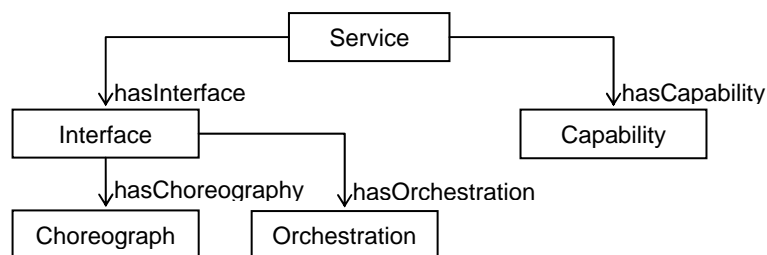


Fig. 2. Main classes used in the description of WSMO Web services.

specifies the desired service. WSMO Capability describes service functionality in terms of the following parameters:

- Preconditions: conditions that should hold for the information space before the web service is performed;
- Post-conditions: set of conditions that will hold for the information space after a successful completion of the web service;
- Assumptions: conditions that should hold for the state of the world before the web service is performed; and
- Effects: set of conditions that will hold in the state of the world after a successful completion of the web service.

Description of input and output types are also part of both goal and web service descriptions in IRS-III. That means, one can describe Web services by their input and output parameters, as well as the above-mentioned parameters described in the capability of WSMO specifications. WSMO does not treat Input and output type description explicitly as part of its Capability description. However, one can specify input or output types of a service as a constraint in WSMO Preconditions or Post-conditions, respectively.

OWL-S takes a different approach to describe Web service functional properties. It describes a service by a process model. The process is then described in terms of its Input, Output, as well as Precondition and Effect expressions (IOPEs). OWL-S functional properties appear in the Profile (as well as the Process description), but only as a replication of IOPEs in the Process. However, OWL-S does not impose any constraint on the consistency of IOPEs in the Profile with those in the Process [10]. The IOPEs presented in the Process description is a more reliable source of knowledge, although, either one could be taken into account. In OWL-S version 1.1 [10], conditional Output and Effect are called Result and a new set of variables called Locals are introduced.

A distinction between the information space and real world state is made by WSMO, which is different from that of OWL-S. An example of a WSMO Precondition for a service can be validity of the credit card of a customer provided by the client agent. This condition can be evaluated in the information space available to the service, for example, by checking the credit card number with another service.

On the other hand an Assumption is a condition in the state of the world that may not be evaluated in the information space available to the service. An example of an Assumption is the size or weight limitation of goods, described for a transportation service. This Assumption cannot be evaluated in the available information space (if it could be evaluated, then it should be defined as a Precondition). Yet, it can be evaluated against a request for a service (i.e. a Goal). Furthermore, the invoker of a service can be informed that the service provider is assuming a condition before invoking the service. In our example above, the invoker would be informed that the service provider assumes the weight of the goods to be in a certain range. OWL-S does not distinguish between the conditions on the state of the world and the information space. However, one may use OWL-S Precondition or Effect to express both types of WSMO conditions.

OWL-S Local parameters are variables that are not provided by the invoker, but submitted by any other means or only being used in logical expressions without any value being assigned to them (such as the variable used for expressing the weight of

goods in the example above). Since WSMO does not treat inputs and output parameters separately from conditions in its Capability descriptions, Locals do not explicitly appear in the WSMO Capability descriptions, either. However there is no restriction to express such parameters and their type definitions in WSMO description.

The Result of a service in OWL-S (i.e. Output and Effect) is bound to a condition. These conditions are introduced to describe a service that may have different outcomes depending on the circumstances. For example, in a book-selling service, two different results are possible. First, a successful purchase that results in:

- charging the credit card (IRS-III and WSMO: Post-condition, OWL-S: Effect);
- shipping the book to the customer's address (IRS-III, WSMO and OWL-S: Effect);
- and
- sending a purchase acknowledgement to the client agent or invoker (IRS and OWL-S: Output, WSMO Post-condition).

Second, a successful reservation for the book that results in:

- reserving a book on the next delivery; and
- sending a reservation acknowledgment.

The above Results will take place depend on a condition: "if the book title is available in the stock". Such condition appears explicitly in OWL-S as part of description of Results. In WSMO, such conditions can appear implicitly as part of Effect or Post-condition description; however they are not stated explicitly.

The two sets of properties to describe functional properties of services, although very different at the first glance, can nearly express the same semantics. However a straight forward mapping between the two is not trivial. WSMO distinguishes between the conditions applied to the information space and the state of the world. On the other hand, OWL-S defines the Local parameters and explicit conditions for result. OWL-S Precondition generalizes Assumption and Precondition, and OWL-S Effect generalizes WSMO Post-condition and Effect. As a result mapping conditions from WSMO service Capability to OWL-S is straight forward, while the other way around requires a classification of the conditions. As IRS-III uses input and output types to describe Web services, their mapping is also straight forward (see Fig. 3).

OWL-S allows any logical language (e.g. SWRL, KIF, etc. see [10] for details) for representing a logical condition, but the language must be specified in their description. WSMO has also been a language neutral specification so far; however

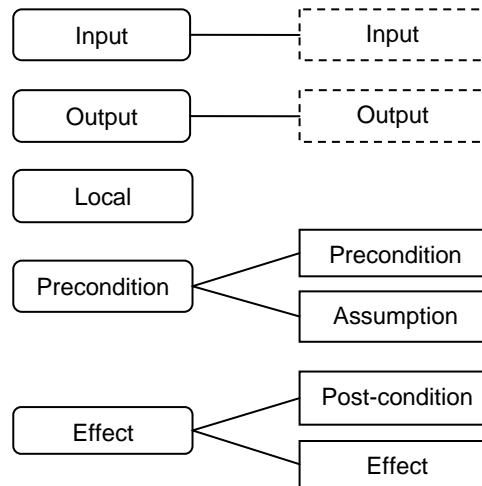


Fig. 3. Similarity between OWL-S IOPEs and WSMO (IRS-III) Capability.

expressions in all WSMO examples and use cases are represented in WSML [16]. As mentioned above IRS-III implementation uses an OCML based inference engine. However, we provide translator for OWL [11] expressions. Currently we are linking the IRS-III API [Domingue et al., 2004] to the WSMO4J API [wsmo4j.sourceforge.net] which has an inbuilt WSML [16] parser.

6 Mediators

The notion of Mediator is specific to WSMO and it is one of its fundamental components. Mediators may be seen as a wrapper for a service or a goal that performs mediation. This wrapper keeps particular specification to describe a Mediator. WSMO Mediators keep the following information:

- Mediation Service: a goal or a service to perform the mediation,
- Source: the entity providing inputs to the mediator,
- Target: where the result of the mediation will be provided to.

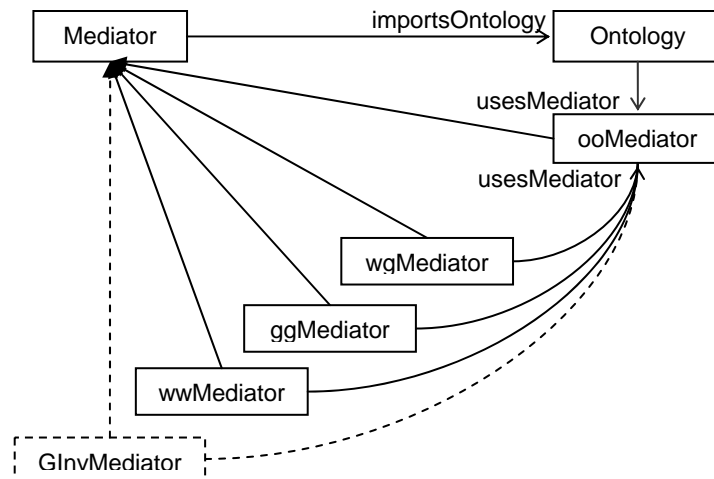


Fig. 4. Different types of mediators in WSMO.

Different types of mediators are seen to mediate data or ontology in interaction between different components. Four types of Mediators are introduced by WSMO as following (Fig. 4) [18]:

- **wgMediator** mediates Web services to Goals. This mediator represents mediation between a Web service and the Goal description that it fulfills.
- **ggMediator** mediates between two Goals. This mediator represents the reduction of the source Goal description into the target Goal.
- **wwMediator** mediates between two Web services.

- **ooMediator** imports ontologies, resolve possible mismatches and find mappings between ontologies. This mediator type can be used by the other three types of mediators to resolve ontological mismatches in the description of their source and target entities.

Apart from the four above mentioned mediators IRS-III introduces a fifth type of mediators, namely Goal Invocation (or GInv) mediator. It is used to mediate the result of a Goal invocation. We used this type of mediators to mediate between components in a composition. (We refer to this type of mediators in Section 7.)

OWL-S does not distinguish between Web services and mediators. In other words, one can define a Web service to perform mediation without explicitly describing it as a mediator in its semantic descriptions. This fact makes the translation of OWL-S to WSMO very difficult. It is not a straight forward process to decide whether a service is actually a mediator and what are the target and source entities of a mediator. The backward translation, however, is simple but causes information loss.

7 Service Composition

Reusability is an important characteristic of Web services that makes them suitable for building composite Web services. A composed service is a more complex service that is built by combining existing services. Web service composition attracted much attention in domain of knowledge representation and Semantic Web, particularly research on automatic service composition. Service composition has also been addressed in domain of Web services (e.g. BPEL4WS [8]), as well as Semantic Web Services. Nevertheless, this section discusses only the issues related to OWL-S and IRS-III composition models as well as existing WSMO Orchestration model [17].

WSMO model for describing composition [17] is still under development. However, the general idea of distinction between data flow and control flow is already described in the available documents. This idea has influenced the models implemented in IRS-III [6]. Orchestration is part of Web Service Interface description. It is to provide the necessary details for the execution of all the component services in a composition.

On the other hand, the OWL-S Process is classified in two subclasses: AtomicProcess and CompositeProcess that represent invocable services (SimpleProcess is discussed in section 8). An AtomicProcess is, in fact, a stateless service that receives a set of inputs to start the process and produces a set of outputs after it is executed. A CompositeProcess is composed of component Processes and it is stateful [4].

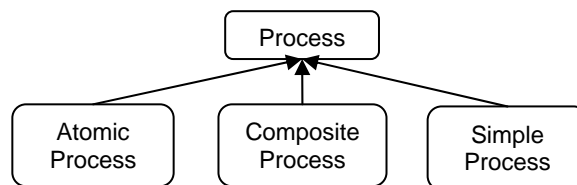


Fig. 5. Subclasses of OWL-S process.

The first difference one may notice between the two descriptions is that WSMO has only one type of Web Service (Fig. 2) and it does not distinguish a composite service from an atomic one, as opposed to OWL-S (Fig. 5). WSMO keeps the composition description as part of its service description (Fig. 6). Essentially, that is one of the ideas in WSMO descriptions to hide implementation complications from the client. It is not the concern of a client if the provided service is composed of several services or it is atomic. In fact in extreme cases, the Orchestration can be a proprietary piece of knowledge for the provider, and therefore, not accessible to others including the client agent. Yet, it is important for a client to know if the service is stateful or stateless; or how to interact with a service. That piece of knowledge is provided in the WSMO Choreography and we discuss that in the next section.

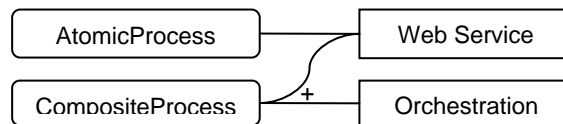


Fig. 6. Mapping of OWL-S Atomic and Composite processes to WSMO Web service.

OWL-S composition model is based on a set of ControlConstructs, such as a Sequence, IfThenElse, Concurrent, etc. to describe the control flow. It follows the structured system design paradigm similar to BPEL4WS [8]. That is, every ControlConstruct is composed of a set of other constructs or processes. Furthermore, OWL-S provides the means to define data bindings between the component processes.

IRS-II [14] offered support for composing Web services where the flow of data and execution had to be defined in its proprietary language, OCML, which provided much flexibility at the cost of high level of complexity. IRS-III, however, presents different models to describe the composition of the Web Services and hides the OCML description from a user or client agent. As mentioned before IRS-III composition models distinguish between control flow and data flow descriptions. We have implemented two major approaches describing compositions, namely structured model [6] and state-based model. They differ mainly in the control flow description, while both use the same model for data flow description. The data flow is modeled using data bindings that are implemented based on the WSMO Mediators (Sec. 6).

The structured composition model in IRS-III is similar to OWL-S composition model, as far as the control flow is concerned. That is a translation of OWL-S ControlConstruct control components in IRS-III is straight forward. At the moment, IRS-III structured composition model [6] supports most OWL-S Control Constructs except, Choice and AnyOrder.

The data binding in IRS-III is described by a directed graph, where edges are representing Goal Invocation Mediators. The Mediators are used to perform required transformation between components in IRS-III. However, OWL-S data bindings do not allow any mediation process in the binding description. Translation of OWL-S bindings to simple Mediators in IRS-III is trivial. On the other hand, one may use an OWL-S service description that describes a mediation service. In such cases,

detecting an OWL-S Process that performs mediation and then translating it to a WSMO Mediator is not trivial.

8 Choreography

WSMO Choreography (Fig. 2) describes the information required to interact with a Web service, such as grounding. Interaction with stateless Web services requires the knowledge that can be found in an interface description such as WSDL. WSMO represents its Grounding as part of the Choreography and OWL-S describes it in association to the Service description (Fig. 1). OWL-S already provides Grounding description for WSDL and the WSMO Grounding is under development.

However, the interaction with stateful services (such as composite services) requires more information than that can be provided by WSDL. A stateful service may receive some inputs after it is started and produces some output before it is finished in different states. An intelligent agent or a programmer should be ultimately able to gain the knowledge of how to interact with a composite Web service. Such knowledge is also to be provided by WSMO Choreography description (also called *business protocol* by BPEL4WS). For example, one should not need to access information in the Web service composition to obtain the information about exchange of the messages with the service. Since a composition may include far more information needed or it may be considered private property of the service provider. Whilst Choreography has been addressed by IRS-III [3] and WSMO [17] (under development), OWL-S does not represent such notion in its specification, yet. The knowledge about interaction with an OWL-S composite service should be obtained from the CompositeProcess description. As a consequence, any future mapping from OWL-S to WSMO Choreography should extract such knowledge from the OWL-S CompositeProcess descriptions.

9 Goal Description

A principle underlying WSMO and IRS-III is the clean ontological separation of user and web service contexts. A goal description thus is used to represent the domain of the client of a web service. The client may be a human user or another web service. The key is that the domain of the client is distinct from the domain of the web service. For example, a user may request to go on holiday with preferences for a warm climate, ancient Greek culture and a child friendly environment whereas holiday based services are based on flight and hotel booking services. Moreover, some user goals may be even in principle unsatisfiable by any computer based web service, for example “I would like to marry Britney Spears”.

So although there are some similarities between OWL-S profiles and WSMO capabilities this does not indicate that OWL-S has a counterpart to goal descriptions as found in WSMO or IRS-III.

10 Conclusion

In this paper we explored main aspects of two major specifications for describing Semantic Web Services, OWL-S and WSMO. As we briefly introduced the major concepts in both, we showed the similarities in both specifications while describing the differences in their perspectives. A thorough mapping between the two that keeps the detailed information while translating the descriptions is by no means trivial. Some notions such as functional parameters (OWL-S IOPE and WSMO Capability) may be translated nearly with no data loss. A tool mainly translating IOPE and Capability is developed and presented in [7].

Two notions of Goal and Mediators are of particular value in WSMO. OWL-S Profile or SimpleProcess can be considered similar to goals depending on the context. Mediators are not explicitly described in OWL-S; however OWL-S service descriptions may implicitly describe a service that performs mediation and that makes translation very difficult.

OWL-S composition descriptions may be mapped to IRS-III composition model, however IRS-III does not support all the features described in OWL-S, yet. Finally, OWL-S does not explicitly describe Choreography of message exchange with a service. For that purpose a translator should extract such information from the OWL-S composition description.

Acknowledgement

This work is supported by the Data, Information and Process Integration (DIP) project funded under the European Union's IST program (FP6-507483); the Advanced Knowledge Technologies (AKT) project funded by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01; and the CNPq, Brazil in form of a scholarship held by Mr. Sell.

References

- [1] S. Decker, et al. "The Semantic Web: The Roles of XML and RDF", IEEE Internet computing, 63-74, 4(5), 2000.
- [2] J. Domingue, L. Cabral, F. Hakimpour, D. Sell and E. Motta, "IRS-III: A Platform and Infrastructure for Creating WSMO-based Semantic Web Services", Proc. of the Workshop on WSMO Implementations, Germany, 2004.
- [3] J. Domingue and S. Galizia, "Towards a Choreography in IRS-III", Proc. of the Workshop on WSMO Implementations, Germany, 2004.
- [4] I. Foster et al., "Modeling Stateful Resources with Web Services", 2004.
- [5] A. Gomez-Perez, R. Gonzalez-Cabero and M. Lama, "ODE: SWS: A framework for Designing and Composing Semantic Web Services", IEEE Intelligent Systems, 19(4), pp. 24-31
- [6] F. Hakimpour, D. Sell, L. Cabral, J. Domingue and E. Motta, "Semantic Web Service Composition in IRS-III: The Structured Approach", to appear in the proc. of the 7th Int'l IEEE Conf. on e-Commerce Technology (IEEE CEC'05), Germany, 2005.

- [7] F. Hakimpour, J. Domingue, E. Motta, L. Cabral and Y. Lei, "Integration of OWL-S into IRS-III", AKT Workshop on Semantic Web Services, AKT-SWS04, 2004.
- [8] IBM Corporation, Business Process Execution Language for Web Services, www-128.ibm.com/developerworks/library/ws-bpel/, 2003.
- [9] R. Lara, A. Polleres, H. Lausen, D. Roman, J. de Bruijn, and D. Fensel, "A Conceptual Comparison between WSMO and OWL-S", Draft WSMO deliverable 4.1, www.wsmo.org/2004/d4/d4.1/
- [10] D. Martin (eds), "OWL-S: Semantic Markup for Web Services", www.daml.org/services/owl-s/1.1/overview/.
- [11] D. L. McGuinness and F. V. Harmelen (eds.), OWL Web Ontology Language Overview. www.w3.org/TR/owl-features/, 2004.
- [12] P. Mika, D. Oberle, A. Gangemi and M. Sabou, "Foundations for Service Ontologies: Aligning OWL-S to DOLCE", WWW2004, USA, ACM, 2004.
- [13] E. Motta, "An Overview of the OCML Modelling Language", the 8th Workshop on Knowledge Engineering Methods and Languages, 1998.
- [14] E. Motta, J. Domingue, L. Cabral and M. Gaspari, IRS-II: A Framework and Infrastructure for Semantic Web Services. 2nd Int'l Semantic Web Conf., 2003.
- [15] B. Omelayenko, M. Crubézy, D. Fensel, Y. Ding, E. Motta, and M. Musen, "Meta Data and UPML (UPML Version 2.0)", www.cs.vu.nl/~upml/upml2.0.pdf, 2000.
- [16] WSML, "Web Service Modeling Language" www.wsmo.org/TR/d16/, 2005
- [17] WSMO, "Ontology-based Choreography and Orchestration of WSMO Services", www.wsmo.org/2004/d14/, 2005.
- [18] WSMO, "Web Service Modeling Ontology-Standard", www.wsmo.org/2004/d2/, 2004.
- [19] WSMO, "WSMO Use Case: Virtual Travel Agency" www.wsmo.org/2004/d3/d3.3/, 2004