Approaches to Semantic Web Services: An Overview and Comparisons

Liliana Cabral[1], John Domingue[1], Enrico Motta[1],
Terry Payne[2] and Farshad Hakimpour[1]

[1] Knowledge Media Institute, The Open University, Milton Keynes, UK
{L.S.Cabral, J.B.Domingue, E.Motta, F.Hakimpour}@open.ac.uk
[2] IAM, University of Southampton, Southampton, UK
trp.@ecs.soton.ac.uk

Abstract. The next Web generation promises to deliver Semantic Web Services (SWS); services that are self-described and amenable to automated discovery, composition and invocation. A prerequisite to this, however, is the emergence and evolution of the Semantic Web, which provides the infrastructure for the semantic interoperability of Web Services. Web Services will be augmented with rich formal descriptions of their capabilities, such that they can be utilized by applications or other services without human assistance or highly constrained agreements on interfaces or protocols. Thus, Semantic Web Services have the potential to change the way knowledge and business services are consumed and provided on the Web. In this paper, we survey the state of the art of current enabling technologies for Semantic Web Services. In addition, we characterize the infrastructure of Semantic Web Services along three orthogonal dimensions: *activities, architecture and service ontology.* Further, we examine and contrast three current approaches to SWS according to the proposed dimensions.

1 Introduction

In recent years, distributed programming paradigms have emerged, that allow generic software components to be developed and shared. Whilst early versions were little more than shared libraries of functions with little user documentation and unpredictable side effects, it wasn't until the advent of object-oriented programming and architectures such as CORBA, that self contained components could be reliably defined, documented and shared within a distributed environment. Although ideal for some enterprise integration and eCommerce, it has only been with the adoption of XML as a common data syntax that the underlying principals have gained wide scale adoption, through the definition of Web Service standards. Web services are well defined, reusable, software components that perform specific, encapsulated tasks via standardized Web-oriented mechanisms. They can be discovered, invoked, and the composition of several services can be choreographed, using well defined workflow modeling frameworks.

Whilst promising to revolutionize eCommerce and enterprise-wide integration, current standard technologies for Web services (e.g. WSDL [6]) provide only syntactic-level descriptions of their functionalities, without any formal definition to what the syntactic definitions might mean. In many cases, Web services offer little more than a formally defined invocation interface, with some human oriented metadata that describes what the service does, and which organization developed it (e.g. through UDDI descriptions). Applications may invoke Web services using a common, extendable communication framework (e.g. SOAP). However, the lack of machine readable semantics necessitates human intervention for automated service discovery and composition within open systems, thus hampering their usage in complex business contexts.

Semantic Web Services (SWS) relax this restriction by augmenting Web services with rich formal descriptions of their capabilities, thus facilitating automated composition, discovery, dynamic binding, and invocation of services within an open environment A prerequisite to this, however, is the emergence and evolution of the Semantic Web, which provides the infrastructure for the semantic interoperability of Web Services. Web Services will be augmented with rich formal descriptions of their capabilities, such that they can be utilized by applications or other services without human assistance or highly constrained agreements on interfaces or protocols. Thus, Semantic Web Services have the potential to change the way knowledge and business services are consumed and provided on the Web.

Current efforts in developing Semantic Web Service infrastructures can be characterized along three orthogonal dimensions: *usage activities, architecture and service ontology*. Usage activities define the functional requirements, which a framework for Semantic Web Services ought to support. The architecture of SWS describes the components needed for accomplishing the activities defined for SWS, whereas the service ontology aggregates all concept models related to the description of a Semantic Web Service.

In this paper we survey the state of the art of current enabling technologies for Semantic Web Services. Further, we examine and contrast three current approaches to SWS according to the proposed dimensions. The rest of the paper is structured as follows: in section 2 we provide a general overview of Web services; in section 3 we provide an overview of the Semantic Web and in particular of those aspects which allow the specification of semantic description for Web services. In section 4 we describe Semantic Web Services according to the dimensions introduced earlier. Sections 5-7 describe the main existing approaches to delivering SWS. Finally we compare and discuss the main differences among the approaches presented.

2 Web services

Web Services are changing the way applications communicate with each other on the Web. They promise to integrate business operations, reduce the time and cost of Web application development and maintenance as well as promote reuse of code over the World Wide Web. By allowing functionality to be encapsulated and defined in a reusable standardized format, Web services have enabled businesses to share (or trade)

functionality with arbitrary numbers of partners, without having to pre-negotiate communication mechanisms or syntax representations. The advent of discovery has enabled vendors to search for Web services, which can then be invoked as necessary. For example, a book-selling company may look for shipping services, which they may later invoke to ensure that books are delivered to the customers. This flexibility is achieved through a set of well-defined standards that define syntax, communication protocol, and invocation signatures, which allow programs implemented on diverse, heterogeneous platforms to interoperate over the internet.

A Web Service is a software program identified by an URI, which can be accessed via the internet through its exposed interface. The interface description declares the operations which can be performed by the service, the types of messages being exchanged during the interaction with the service, and the physical location of ports, where information should be exchanged. For example, a Web service for calculating the exchange rate between two money currencies can declare the operation *getExchangeRate* with two inputs of type string (for source and target currencies) and an output of type float (for the resulting rate). A binding then defines the machine and ports where messages should be sent. Although there can be many ways of implementing Web services, we basically assume that they are deployed in Web servers such that they can be invoked by any Web application or Web agent independently of their implementations. In addition Web services can invoke other Web services.

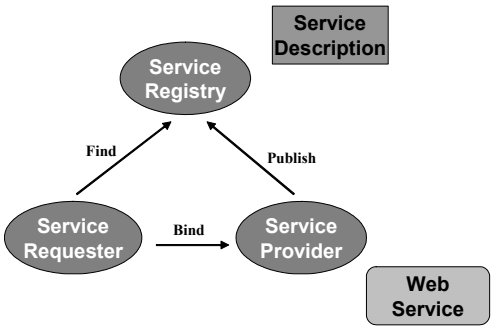

Fig. 1. Web Service usage scenario.

The common usage scenario for Web services (fig. 1) can be defined by three phases; *Publish, Find,* and *Bind;* and three entities: the service requester, which invokes services; the service provider which responds to requests; and the registry where services can be published or advertised. A service provider *publishes* a description of a service it provides to a service registry. This description (or *advertisement*) includes a profile on the provider of the service (e.g. company name and address); a profile about the service itself (e.g. name, category); and the URL of its service interface definition (i.e. WSDL description).

When a developer realizes a need for a new service, he *finds* the desired service either by constructing a query, or browsing the registry. The developer then interprets

the meaning of the interface description (typically through the use of meaningful label or variable names, comments, or additional documentation) and *binds* to (i.e. includes a call to invoke) the discovered service within the application they are developing. This application is known as the *service requester*. At this point, the service requester can automatically invoke the discovered service (provided by the service provider) using Web service communication protocols (i.e. SOAP).

Key to the interoperation of Web services is an adoption of a set of enabling standard protocols. Several XML-based standards (fig. 2) have been proposed to support the usage scenario previously described.

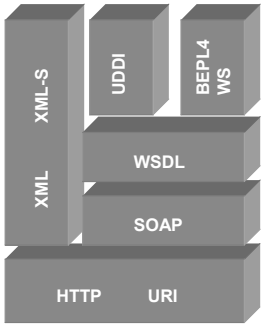

Fig. 2. Web Services enabling standards

XML schema (XML-S) [2] provides the underlying framework for both defining the Web Services Standards, and variables/objects/data types etc that are exchanged between services. SOAP [27] is W3C's recommended XML-data transport protocol, used for data exchange over web-based communications protocols (http). SOAP messages can carry an XML payload defined using XML-S, thus ensuring a consistent interpretation of data items between different services.

WSDL [6] is the W3C recommended language for describing the service interface. Two levels of abstraction are used to describe Web services; the first defines atomic method calls, or *operations*, in terms of input and output *messages* (each of which contain one or more parameters defined in XML-S). Operations define the way in which messages are handled e.g. whether an operation is a *one-way operation, request-response, solicit-response* or *notification*. The second abstraction maps operations and associated messages to physical endpoints, in terms of *ports* and *bindings*. Ports declare the operations available with corresponding inputs and outputs. The bindings declare the transport mechanism (usually SOAP) being used by each operation. WSDL also specifies one or more network locations or endpoints at which the service can be invoked.

As services become available, they may be registered with a UDDI registry [26] which can subsequently be browsed and queried by other users, services and applications. UDDI web service discovery is typically human oriented, based upon yellow or white-page queries (i.e. metadata descriptions of service types, or information about the service providers). UDDI service registrations may also include references to

WSDL descriptions, which may facilitate limited automation of discovery and invocation. However, as no explicit semantic information is normally defined, automated comprehension of the WSDL description is limited to cases where the provider and requester assume pre-agreed ontologies, protocols and shared knowledge about operations.

A service might be defined as a workflow describing the choreography of several operations. Such a workflow may determine: the order of operation execution; what operations may be executed concurrently; and alternative execution pathways (if conditional operators are included in the workflow modeling language). Conversely, workflows are required to orchestrate the execution of several simple services that may be composed together for forming a more complex service. Various choreography and orchestration languages have been proposed such as BPEL4WS [5], and are currently being evaluated by various industry standardization bodies.

3 The Semantic Web

The Semantic Web is a vision of a Web of meaningful contents and services, which can be interpreted by computer programs (see for example [1]). It can also be seen as a vast source of information, which can be modelled with the purpose of sharing and reusing knowledge. Semantic Web users will be able to do more accurate searches of the information and the services they need from the tools provided.

The Semantic Web provides the necessary infrastructure for publishing and resolving ontological descriptions of terms and concepts. In addition, it provides the necessary techniques for reasoning about these concepts, as well as resolving and mapping between ontologies, thus enabling semantic interoperability of Web Services through the identification (and mapping) of semantically similar concepts.

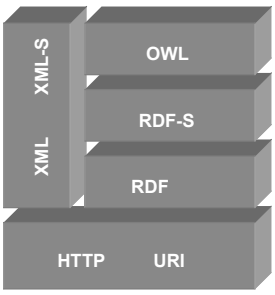

Fig. 3. Semantic Web Enabling standards

Ontologies have been developed within the Knowledge Modelling research community [11] in order to facilitate knowledge sharing and reuse. They provide greater expressiveness when modelling domain knowledge and can be used to communicate this knowledge between people and heterogeneous and distributed application systems.

As with Web Services, Semantic Web enabling standards fit into a set of layered specifications (fig. 3) built on the foundation of URIs and XML Schema. The current components of the Semantic Web framework are RDF [13], RDF Schema (RDF-S) [3] and the Web Ontology Language – OWL [4]. These standards build up a rich set of constructs for describing the semantics of online information sources.

RDF is a XML-based standard from W3C for describing resources on the Web. RDF introduces a little semantics to XML data by allowing the representation of objects and their relations through properties. RDF-Schema is a simple type system, which provides information (metadata) for the interpretation of the statements given in RDF data. The Web Ontology language – OWL will facilitate greater machine interpretability of Web content than RDF and RDF Schema by providing a much richer set of constructs for specifying classes and relations. OWL has evolved from existing ontologies languages and specifically from DAML+OIL [12].

4 Semantic Web Services

Semantic descriptions of Web services are necessary in order to enable their automatic discovery, composition and execution across heterogeneous users and domains. Existing technologies for Web services only provide descriptions at the syntactic level, making it difficult for requesters and providers to interpret or represent non-trivial statements such as the meaning of inputs and outputs or applicable constraints. This limitation may be relaxed by providing a rich set of semantic annotations that augment the service description. A Semantic Web Service is defined through a service ontology, which enables machine interpretability of its capabilities as well as integration with domain knowledge.

The deployment of Semantic Web Services will rely on the further development and combination of Web Services and Semantic Web enabling technologies. There exist several initiatives (e.g. http://dip.semanticweb.org or http://www.swsi.org) taking place in industry and academia, which are investigating solutions for the main issues regarding the infrastructure for SWS.

Semantic Web Service infrastructures can be characterized along three orthogonal dimensions (fig. 4): *usage activities, architecture and service ontology*. These dimensions relate to the requirements for SWS at business, physical and conceptual levels. Usage activities define the functional requirements, which a framework for Semantic Web Services ought to support. The architecture of SWS defines the components needed for accomplishing these activities. The service ontology aggregates all concept models related to the description of a Semantic Web Service, and constitutes the knowledge-level model of the information describing and supporting the usage of the service.

From the usage activities perspective, SWS are seen as objects within a business application execution scenario. The activities required for running an application using SWS include: publishing, discovery, selection, composition, invocation, deployment and ontology management, as described next.

The publishing or advertisement of SWS will allow agents or applications to discover services based on its goals and capabilities. A semantic registry is used for registering instances of the service ontology for individual services. The service ontology distinguishes between information which is used for matching during discovery and that used during service invocation. In addition, domain knowledge should also be published or linked to the service ontology.

The discovery of services consists of a semantic matching between the description of a service request and the description of published service. Queries involving the service name, input, output, preconditions and other attributes can be constructed and used for searching the semantic registry. The matching can also be done at the level of tasks or goals to be achieved, followed by a selection of services which solves the task. The degree of matching can be based on some criteria, such as the inheritance relationship of types. For example, an input of type Professor of a provided service can be said to match an input of type Academic of a requested service.

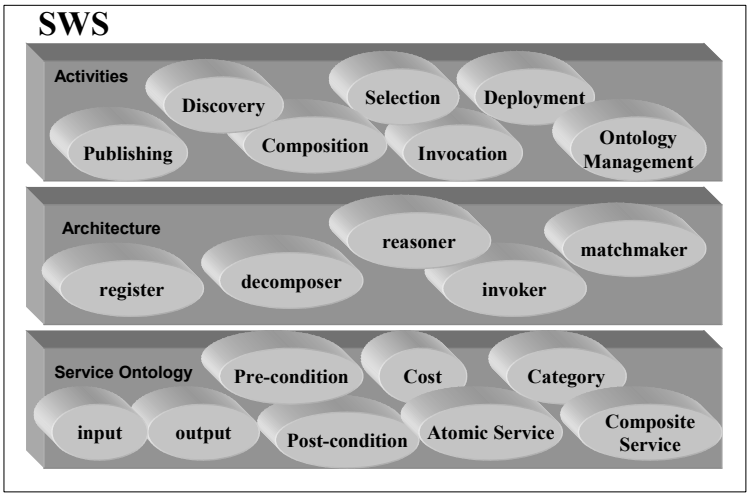

Fig. 4. Semantic Web Services infrastructure dimensions.

A selection of services is required if there is more than one service matching the request. Non-functional attributes such as cost or quality can be used for choosing one service. In a more specialized or agent-based type of interaction a negotiation process can be started between a requester and a provider, but that requires that the services themselves be knowledge-based. In general, a broker would check that the preconditions of tasks and services are satisfied and prove that the services post-conditions and effects imply goal accomplishment. An explanation of the decision-making process should also be provided.

Composition or choreography allows SWS to be defined in terms of other simpler services. A workflow expressing the composition of atomic services can be defined in the service ontology by using appropriate control constructs. This description would

be grounded on a syntactic description such as BEPL4WS [5]. Dynamic composition is also being considered as an approach during service request in which the atomic services required to solve a request are located and composed on the fly. That requires an invoker which matches the outputs of atomic services against the input of the requested service.

The invocation of SWS involves a number of steps, once the required inputs have been provided by the service requester. First, the service and domain ontologies associated with the service must be instantiated. Second, the inputs must be validated against the ontology types. Finally the service can be invoked or a workflow executed through the grounding provided. Monitoring the status of the decomposition process and notifying the requester in case of exceptions is also important.

The deployment of a Web service by a provider is independent of the publishing of its semantic description since the same Web service can have serve multiple purposes. But, the SWS infrastructure can provide a facility for the instant deployment of code for a given semantic description.

The management of service ontologies is a cornerstone activity for SWS since it will guarantee that semantic service descriptions are created, accessed and reused within the Semantic Web.

From the architecture perspective (fig. 4), SWS are defined by a set of components which realize the activities above, with underlying security and trust mechanisms. The components gathered from the discussion above include: a register, a reasoner, a matchmaker, a decomposer and an invoker.

The reasoner is used during all activities and provides the reasoning support for interpreting the semantic descriptions and queries. The register provides the mechanisms for publishing and locating services in a semantic registry as well as functionalities for creating and editing service descriptions. The matchmaker will mediate between the requester and the register during the discovery and selection of services. The decomposer is the component required for executing the composition model of composed services. The invoker will mediate between requester and provider or decomposer and provider when invoking services. These components are illustrative of the required roles in the SWS architecture for the discussion here as they can have different names and a complexity of their own in different approaches.

The service ontology is another dimension under which we can define SWS (fig. 4), for it represents the capabilities of a service itself and the restrictions applied to its use. The service ontology essentially integrates at the knowledge-level the information which has been defined by Web services standards, such as UDDI and WSDL with related domain knowledge. This would include: functional capabilities such as inputs, output, pre-conditions and post-conditions; non-functional capabilities such as category, cost and quality of service; provider related information, such as company name and address; task or goal-related information; and domain knowledge defining, for instance, the type of the inputs of the service. This information can, in fact be divided in several ontologies. However, the service ontology used for describing SWS will rely on the expressivity and inference power of the underlying ontology language supported by the Semantic Web.

Three main approaches have been driving the development of Semantic Web Service frameworks: IRS-II [17], OWL-S [19] and WSMF [9]. IRS-II (Internet Reasoning Service) is a knowledge-based approach to SWS, which evolved from research on

reusable knowledge components [16]. OWL-S is an agent-oriented approach to SWS, providing fundamentally an ontology for describing Web service capabilities. WSMF (Web Service modeling framework) is a business-oriented approach to SWS, focusing on a set of e-commerce requirements for Web Services including trust and security. The following sections describe these approaches in more detail.

5 IRS approach

The Internet Reasoning Service - IRS-II [17] is a Semantic Web Services framework, which allows applications to semantically describe and execute Web services.

IRS-II is based on the UPML (Unified Problem Solving Method Development Language) framework [18], which distinguishes between the following categories of components specified by means of an appropriate ontology:

- *Domain models.* These describe the domain of an application (e.g. vehicles, a medical disease).
- *Task models.* These provide a generic description of the task to be solved, specifying the input and output types, the goal to be achieved and applicable preconditions.
- *Problem Solving Methods (PSMs).* These provide abstract, implementation-independent descriptions of reasoning processes which can be applied to solve tasks in a specific domain.
- *Bridges.* These specify mappings between the different model components within an application.

The main components of the IRS-II architecture are the IRS-II Server, the IRS-II Publisher and the IRS-II Client, which communicate through the SOAP protocol. The IRS-II server holds descriptions of Semantic Web Services at two different levels. A knowledge level description is stored using the UPML framework of tasks, PSMs and domain models. These are currently represented internally in OCML [16], an Ontolingua-derived language which provides both the expressive power to express task specifications and service competencies, as well as the operational support to reason about these. In addition, IRS-II has a special-purpose mapping mechanism to ground competence specifications to specific Web services.

The IRS-II Publisher plays two roles in the IRS-II architecture. Firstly, it links Web services to their semantic descriptions within the IRS-II server. Note that each PSM is associated with exactly one Web service although a Web service may map onto more than one PSM since a single piece of code may serve more than one function. Secondly, the publisher automatically generates a wrapper which turns the code into a Web service. Once this code is published within the IRS-II it appears as a standard message-based Web service, that is, a Web service endpoint is automatically generated. There can be more than one type of Publisher or publishing platform, depending on the implementation of the service. This design option allows for the instant deployment of code during publishing as explained before and mediation between the server and the actual service (code) during invocation.

A key feature of IRS-II is that Web service invocation is capability driven. The IRS-II supports this by providing a task centric invocation mechanism. An IRS-II user simply asks for a task to be achieved and the IRS-II broker locates an appropriate PSM and then invokes the corresponding Web service.

IRS-II was designed for ease of use. Developers can interact with IRS-II through the IRS-II browser, which facilitates navigation of knowledge models registered in IRS-II as well as the editing of service descriptions, the publishing and the invocation of individual services. Application programs can be integrated with IRS-II by using the Java API. These programs can then combine tasks that can be achieved within an application scenario.

6 OWL-S approach

OWL-S (previously DAML-S [9]) consists of a set of ontologies designed for describing and reasoning over service descriptions. OWL-S approach originated from an AI background and has previously been used to describe agent functionality within several Multi-Agent Systems as well as with a variety of planners to solve higher level goals.

OWL-S combines the expressivity of description logics (in this case OWL) and the pragmatism found in the emerging Web Services Standards, to describe services that can be expressed semantically, and yet grounded within a well defined data typing formalism. It consists of three main upper ontologies: the Profile, Process Model and Grounding. The Profile is used to describe services for the purposes of discovery; service descriptions (and queries) are constructed from a description of functional properties (i.e. inputs, outputs, preconditions, and effects - IOPEs), and non-functional properties (human oriented properties such as service name, etc, and parameters for defining additional meta data about the service itself, such as concept type or quality of service). In addition, the profile class can be subclassed and specialized, thus supporting the creation of profile taxonomies which subsequently describe different classes of services.

OWL-S process models describe the composition or orchestration of one or more services in terms of their constituent processes. This is used both for reasoning about possible compositions (such as validating a possible composition, determining if a model is executable given a specific context, etc) and controlling the enactment/invocation of a service. Three process classes have been defined: the composite, simple and atomic process. The atomic process is a single, black-box process description with exposed IOPEs. Inputs and outputs relate to data channels, where data flows between processes. Preconditions specify facts of the world that must be asserted in order for an agent to execute a service. Effects characterize facts that become asserted given a successful execution of the service, such as the physical side-effects that the execution the service has on the physical world. Simple processes provide a means of describing service or process abstractions – such elements have no specific binding to a physical service, and thus have to be realized by an atomic process (e.g. through service discovery and dynamic binding at run-time), or expanded into a composite process. Composite processes are hierarchically defined workflows, consisting of atomic, simple and other composite processes. These process workflows are constructed using

a number of different composition constructs, including: Sequence, Unordered, Choice, If-then-else, Iterate, Repeat-until, Repeat-while, Split, and Split+join.

The profile and process models provide semantic frameworks whereby services can be discovered and invoked, based upon conceptual descriptions defined within Semantic Web (i.e. OWL) ontologies. The grounding provides a pragmatic binding between this concept space and the physical data/machine/port space, thus facilitating service execution. The process model is mapped to a WSDL description of the service, through a thin grounding. Each atomic process is mapped to a WSDL operation, and the OWL-S properties used to represent inputs and outputs are grounded in terms of XML data types. Additional properties pertaining to the binding of the service are also provided (i.e. the IP address of the machine hosting the service, and the ports used to expose the service).

7 WSMF approach

The Web Service Modeling Framework (WSMF) [9] provides a model for describing the various aspects related to Web services. Its main goal is to fully enable e-commerce by applying Semantic Web technology to Web services. WSMF is the product of research on modelling of reusable knowledge components [10].

WSMF is centered on two complementary principles: a strong de-coupling of the various components that realize an e-commerce application; and a strong mediation service enabling Web services to communicate in a scalable manner. Mediation is applied at several levels: mediation of data structures; mediation of business logics; mediation of message exchange protocols; and mediation of dynamic service invocation.

WSMF consists of four main elements: ontologies that provide the terminology used by other elements; goal repositories that define the problems that should be solved by Web services; Web services descriptions that define various aspects of a Web service; and mediators which bypass interoperability problems.

WSMF implementation has been assigned to two main projects: Semantic Web enabled Web Services (SWWS) [25]; and WSMO (Web Service Modelling Ontology) [28]. SWWS will provide a description framework, a discovery framework and a mediation platform for Web Services, according to a conceptual architecture. WSMO will refine WSMF and develop a formal service ontology and language for SWS.

WSMO service ontology includes definitions for goals, mediators and web services. A web service consists of a capability and an interface. The underlying representation language for WSMO is F-logic. The rationale for the choice of F-logic is that it is a full first order logic language that provides second order syntax while staying in the first order logic semantics, and has a minimal model semantics. The main characterizing feature of the WSMO architecture is that the goal, web service and ontology components are linked by four types of mediators as follows:

- OO mediators link ontologies to ontologies,
- WW mediators link web services to web services,
- WG mediators link web services to goals, and finally,
- GG mediators link goals to goals.

Since within WSMO all interoperability aspects are concentrated in mediators the provision of different classes of mediators based on the types of components connected facilitates a clean separation of the different mediation functionalities required when creating WSMO based applications.

8 SWS approaches comparison

This comparison discusses the delivered results of IRS-II, OWL-S and WSMF (SWWS) as they represent the main approaches driving the implementation of Semantic Web Service components. The following table shows the high-level elements of each approach as implemented by the time of this writing fitting into the previously discussed dimensions of SWS, including the application tools provided as well.

Table 1. Delivered components of current SWS approaches

	IRS-II	OWL-S	WSMF
SWS Activities	Publishing Selection Task Achievement	Composition Discovery Invocation	Discovery
Architecture	Server Publisher Client	Daml-s Virtual Machine Matchmaker	Service Registry Profile Crawler
Service Ontology	Task/PSM Ontology	OWL-S	WSMO
Application tools	IRS Browser and Editor; Publisher; Java API	WSDL2DAML-S	Query interface

The IRS-II approach has concentrated efforts in delivering an infrastructure that users can easily use from the stage where they have some service code available, to the semantic markup and publishing of this code, to the invocation of this code through task achievement. Because services are considered atomic in IRS-II, there is no semantic description of composed services, although a PSM can embody a control flow for subtasks. Also, a selection of services is performed for finding which PSMs can solve the task requested.

The service ontology of IRS-II consists of a Task ontology and a PSM ontology, which separate the description of what a service does from the parameters and constraints of a particular implementation. Additionally, the task ontology can also include a domain ontology. In IRS, service constraints (e.g. pre-conditions and post-conditions) must be expressed in OCML but an OWL-to-OCML parser has recently been completed. An import/export mechanism for OWL-S service descriptions, which includes the adoption of the properties of the OWL-S Profile is being implemented as well.

The main contribution of the OWL-S approach is its service ontology, which builds on the Semantic Web stack of standards. OWL-S models capabilities required

for Web services to the extent of grounding, which maps to WSDL descriptions. Additionally, the Daml consortium has put a lot of effort in representing the interactions among Web Services through the process model of the OWL-S service ontology.

Since the OWL-S service ontology is public and does not prescribe a framework implementation it has been used as the starting point of individual efforts towards SWS, for example [15]. Nevertheless, the DAML consortium has implemented some components of an architecture based on the DAML inference engine [20] [21]. The invocation activity of OWL-S involves a decomposition of the process model. The discovery activity demonstrated in [22] relies on the extension of UDDI registry.

The WSMF approach, although delivering a conceptual framework, invested considerable effort in bringing business requirements into account when proposing a conceptual architecture. Some of the outcomes are still in the form of more detailed specifications. In particular, a service registry has been proposed for which a high-level query language is defined according to the service ontology [25]. WSMO distinguished characteristic is the inclusion of mediators in the ontology specification.

In common with IRS-II, the WSMF approach builds on the UPML framework, taking advantage of the separation of tasks (goals) specifications from the service specifications.

9 Discussion and Conclusions

A complete solution for delivering Semantic Web Services is on the way. Although the vision for SWS has been set and many partial solution cases demonstrated (see for example ISWC 2003) for solving particular issues, only now is the area as a whole taking shape. This is evidenced by the fast-paced evolution of the underlying standards and technologies and the proof-of-concept stage of research in the area.

The state of the art of SWS shows that technologies will shape towards accepted enabling standards for Web Services and the Semantic Web. In particular, IRS-II, OWL-S and WSMF promise inter-compatibility in terms of OWL-based service descriptions and WSDL-based grounding.

However, an assessment of the delivered results of IRS-II, OWL-S and WSMF approaches show that Semantic Web Services are far from mature. While they represent different development approaches converging to the same objective, they provide different reasoning support, which are based on different logic and ontology frameworks. Furthermore, they emphasize different ontology-based service capabilities and activities according to the orientation of their approaches.

None of the approaches described provide a complete solution according to the dimensions illustrated, but interestingly enough they show complementary strengths. For example, IRS-II has strong user and application integration support while OWL-S provides a rich XML-based service-ontology. WSMF has a comprehensive conceptual architecture, which covers requirements of one of the most demanding web-based application area, namely e-commerce. These requirements reflect the way business clients buy and sell services.

Summarizing, Semantic Web Services are an emerging area of research and currently all the supporting technologies are still far from the final product. There are

technologies available for creating distributed applications which rely on the execution of Web services deployed on the WWW, however, these technologies require a human user in the loop for selecting services available in registries. Semantic Web technology can be utilised to do the markup and reasoning of Web service capabilities.

We have described the current main approaches to Semantic Web Services: IRS-II, OWL-S and WSMF. These approaches are complementary in many ways and can be compared according to different dimensions of SWS.

Nevertheless, there are still a number of issues concerning Semantic Web Services being investigated in a number of initiatives. These issues range from service composition to service trust and will have the attention of industry and academia for the next few years.

References

1. Berners-Lee, T. Hendler, J., Lassila, O.: The Semantic Web. Scientific American, Vol. 284 (4). (2001) 34-43
2. Biron, P. V., Malhotra, A. (eds.): XML Schema Part 2: Datatypes, W3C Recommendation, 2 May 2001. http://www.w3.org/TR/xmlschema-2/. (2001)
3. Brickley D., Guha R.V. (eds.): RDF Vocabulary Description Language 1.0: RDF Schema, W3C Proposed Recommendation (work in progress). http://www.w3.org/TR/rdf-schema/. (2003)
4. Bechhofer, S., Dean, M., Van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D., Patel-Schneider, P., Schreiber, G., Stein, L.: OWL Web Ontology Language Reference, W3C Proposed Recommendation (work in progress). http://www.w3.org/TR/owl-ref/. (2003)
5. BPEL4WS Consortium. Business Process Execution Language for Web Services. http://www.ibm.com/developerworks/library/ws-bpel
6. Christensen, E. Curbera, F., Meredith, G., Weerawarana, S. Web Services Description Language (WSDL), W3C Note 15. http://www.w3.org/TR/wsdl. (2001)
7. Christoph, B., Fensel, D., Maedche, A.: A Conceptual Architecture for Semantic Web Enabled Web Services. . http://swws.semanticweb.org/public_doc/D2.1.pdf. (2003)
8. DAML-S Coalition: DAML-S 0.9 Draft Release. http://www.daml.org/services/daml-s/0.9/. (2003)
9. Fensel, D., Bussler, C. The Web Service Modeling Framework WSMF. Eletronic Commerce: Research and Applications. Vol. 1. (2002). 113-137
10. Fensel, D. and Motta, E.: Structured Development of Problem Solving Methods. IEEE Transactions on Knowledge and Data Engineering, Vol. 13(6). (2001). 913-932.
11. Gruber, T. R. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2). (1993)
12. Joint US/EU ad hoc Committee. Reference Description of the DAML-OIL Ontology Markup Language. http://www.daml.org/2001/03/reference. (2001)
13. Klyne, G., D., Carroll, J.J. (eds.): Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Proposed Recommendation (work in progress). http://www.w3.org/TR/rdf-concepts/. (2003)
14. Mandell, D., McIlraith, S. Grounding the Semantic Web: A Bottom-up Approach to Automating Web Service Discovery, Customization and Semantic Translation. In

Workshop on E-Services and the Semantic Web (ESSW03) in conjunction with WWW03.
15. McIlraith, S., Son, T. C., Zeng, H. Semantic Web Services. IEEE Intelligent Systems, Vol. 16(2). (2001) 46-53.
16. Motta E.. Reusable Components for Knowledge Modelling. IOS Press, Amsterdam, The Netherlands. (1999)
17. Motta, E., Domingue, J., Cabral, L., Gaspari, M.: IRS-II: A Framework and Infrastructure for Semantic Web Services. In: Fensel, D., Sycara, K., Mylopoulos, J. (volume eds.): The SemanticWeb - ISWC 2003. Lecture Notes in Computer Science, Vol. 2870. Springer-Verlag, Heidelberg (2003) 306–318
18. Omelayenko, B., Crubezy, M., Fensel, D., Benjamins, R., Wielinga, B., Motta, E., Musen, M., Ding, Y..: UPML: The language and Tool Support for Makiing the Semantic Web Alive. In: Fensel, D. et al. (eds.): Spinning the Semantic Web: Bringing the WWW to its Full Potential. MIT Press (2003) 141–170
19. OWL-S Coalition: OWL-S 1.0 Release. http://www.daml.org/services/owl-s/1.0/. (2003)
20. Paolucci, M., Sycara, K. and Kawamura, T.: Delivering Semantic Web Services. Tech. report CMU-RI-TR-02-32, Robotics Institute, Carnegie Mellon University, May, 2003
21. Paolucci, M., Ankolekar, A., et al.: The Daml-S Virtual Machine. In: Fensel, D., Sycara, K., Mylopoulos, J. (volume eds.): The Semantic Web - ISWC 2003 Proceedings. Lecture Notes in Computer Science, Vol. 2870. Springer-Verlag, Heidelberg (2003) 290-305
22. Paolucci, M., Kawamura, T., Payne, T., Sycara, K.: Semantic Matching of Web Services Capabilities. In: Horrocks, I. Handler, J. (eds.): The Semantic Web - ISWC 2002 Proceedings. Lecture Notes in Computer Science, Vol. 2342. Springer-Verlag, Heidelberg (2002) 333-347
23. Sirin, E., Hendler, J. and Parsia, B. Semi-automatic Composition of Web Services using Semantic Descriptions. In: Web Services: Modeling, Architecture and Infrastructure workshop in conjunction with ICEIS2003. (2003).
24. Wu, D., Parsia, B., et al: Automating DAML-S Web Services Composition Using SHOP2. In: Fensel, D., Sycara, K., Mylopoulos, J. (volume eds.): The SemanticWeb - ISWC 2003. Lecture Notes in Computer Science, Vol. 2870. Springer-Verlag, Heidelberg (2003) 195-210
25. SWWS Consortium. Report on Development of Web Service Discovery Framework. October 2003. http://swws.semanticweb.org/public_doc/D3.1.pdf
26. UDDI Consortium. UDDI Specification. http://www.uddi.org/specification.html (2000)
27. W3C. SOAP 1.2, W3C Recommendation. http://www.w3.org/TR/soap12-part0/ (2003)
28. WSMO Working Group. Web Service Modelling Ontology Project. DERI Working Drafts. http://www.nextwebgeneration.org/projects/wsmo/ (2004)