**Coverpage**

# Semantic Web Service Architecture —
# Evolving Web Service Standards toward the Semantic Web

 **Tanja Sollazzo, Siegfried Handschuh, Steffen Staab, Martin Frank**

**Abstract.**
The importance of Web services has been recognized and widely accepted by industry and academic research. However, the two worlds have proposed solutions that progress along different dimensions. Academic research has been mostly concerned with expressiveness of service descriptions, while industry has focused on modularization of service layers — mostly for usability in the short term. This paper is concerned with merging the two streams of progress. Our point of departure is the current proposal by IBM. Its proposal is extended by Semantic Web technologies such that a smooth evolution from Web services in the current Web to Web services in the Semantic Web appears possible and — in fact — highly desirable. As a showcase we describe SWOBIS, an ontology-compatible registry for software tools, that is open to usage in the Semantic Web.

# Semantic Web Service Architecture —
# Evolving Web Service Standards toward the Semantic Web

## Abstract

The importance of Web services has been recognized and widely accepted by industry and academic research. However, the two worlds have proposed solutions that progress along different dimensions. Academic research has been mostly concerned with expressiveness of service descriptions, while industry has focused on modularization of service layers — mostly for usability in the short term. This paper is concerned with merging the two streams of progress. Our point of departure is the current proposal by IBM. Its proposal is extended by Semantic Web technologies such that a smooth evolution from Web services in the current Web to Web services in the Semantic Web appears possible and — in fact — highly desirable. As a showcase we describe SWOBIS, an ontology-compatible registry for software tools, that is open to usage in the Semantic Web.

## Introduction

Web services are "self-contained, self-describing modular applications" (Martin 2001). They constitute software modules that "describe a collection of operations that are network-accessible through standardized XML messaging" (Kreger 2001, p. 6). With UDDI, SOAP, .NET and WSDL, industry has made a bold move forward and started initiatives that target the potential benefits of Web services. The focus of the initiatives was — rather naturally — an evolutionary step from current Web technology toward a technology for Web services. Key concerns of the intiatives were, e.g., short-term applicability or scalability. This implies that the corresponding Web service architectures build on little really new technology inside, e.g. they use standardized taxonomies and vocabularies that exhibit little flexibility and expressiveness and that restricts the usability of Web services mostly to human users rather than machine agents. For the latter one would need, e.g., Web service description languages that support semi-structured data, constraints, types and inheritance.

In contrast to the industry point of view, academic research has investigated languages that fulfill exactly these needs (Horrocks *et al.* 2001; Ankolenkar *et al.* 2001; Fensel *et al.* 1999) offering extensible ontology frameworks and layering of languages in the Semantic Web. To the detriment of the latter community the adoption of their schemes into industry (quasi-)standards for Web services is far from trivial, because there is no coherent architecture that allows for pointing out immediate benefits.

The core idea of this paper is to present such an architecture that combines the two worlds and their potential benefits. The benefits of the integration include increased visibility of Web services, because open ontology frameworks allow for semantically expressive advertising on the Web that may be found by Web crawlers. They include better usability because of more expressive Web service descriptions. They include a smooth evolution from Web services for human users such as targeted by current industry (quasi-)standards toward Web services for personalized machine agents that assist the user.

The structure of this paper is as follows. First, we sketch the general model of the Web service setting, the architecture of IBM in particular. We consider the latter the most elaborate and the best described industry quasi-standard for Web Services so far[1]. Second, we analyse assumptions of this (and related) architecture(s), describing several parameters which may be varied to turn the "traditional" view into a Semantic Web view. Third, we outline the current description of DAML-S (Ankolenkar *et al.* 2001). Fourth, we critically evaluate the achievements of DAML-S. Fifth, we describe the integrated architecture that we propose. Finally, we sketch SWOBIS, an ontology-based registry for software tools that we are currently developing into the direction of our proposed integrated architecture.

## IBM Web Service Architecture

The typical procedure in a Web service setting is the following: A Web service provider offers services on the Web. He may choose to register her service at an online registry of a service broker (Trastour & Bartolini 2001, p.2/3). The registry publishes and locates services. To allow for service discovery, the registry also provides standardized description facilities, e.g. taxonomies that allow the description of, *(i)*, the functionality of a service, *(ii)*, its service provider, *(iii)*, how how to access and interact with the service. The

---

[1]The authors are not affiliated with IBM nor its partner companies.

corresponding information about a particular service is registered by the provider at a broker.

The (human) requestor searches for a service at the registry. It finds one by browsing or querying the registry. Then she uses the service description to create a binding for her application to be able to invoke or interact with the Web service implementation. In the IBM Web service architecture this procedure boils down to the architecture layers depicted in Figure 1 (cf. (Kreger 2001)).

**Network Protocols and SOAP.** The upper layers are based on general network protocols, like HTTP or FTP, and SOAP (simple object access protocol[2]), which is a standardized enveloping mechanism for communicating document-centric messages and remote procedure calls using XML.
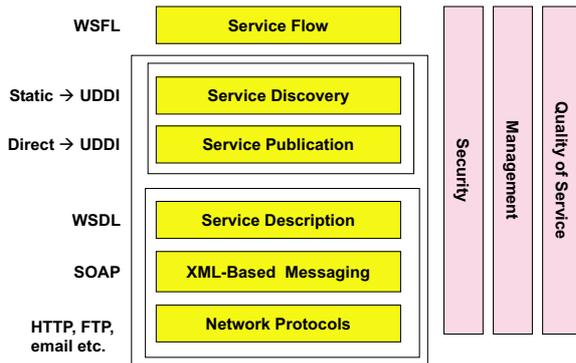


Figure 1: IBM Web Service Architecture

**WSDL.** The interface of a Web service and its service interactions are described by Web Services Description Language (WSDL[3]). WSDL allows to compose XML documents for in order to describe technical details of a Web service.

**UDDI.** The *service publication* is regarded as a business-related description of a service; *e.g.* answering questions such as: What products are associated with this service? Which organization is offering this service? Universal Description Discovery and Integration (UDDI), initiated by Microsoft, IBM, and Ariba[4], does not only consist of defining a data structure standard for all business-related descriptions of services (i.e. service publication), but it also contains all mechanisms that allow the service requestor to gain access to the service publication and service description, thus is also in charge of the *service discovery* layer. For the latter purpose, a UDDI registry contains data in a combination of white pages, yellow pages, and green pages. The information of the white pages includes items such as a company's name, its address, and other contact information. The yellow pages help categorize a business; they define its business type, as well as the industry in which it operates. Finally, the green pages define what kind of services a business offers, and how to electronically communicate with these services.

---

[2]http://www.w3.org/TR/SOAP/

[3]http://www.w3.org/TR/wsdl.html

[4]http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf

**WSFL.** Web Services Flow Language (WSFL) prescribes an XML format for specifying service composition (also called service flow) (Leymann 2001). WSFL allows to compose complex services from given simpler ones.

**Remainder.** Several layers are orthogonal in this architecture, because issues like security, management and quality of service span over all the other layers.

## Critique of the IBM (Quasi-)Standard Architecture

This desription, the current industry quasi-standard, however, is too "flat" to be comprehensive. Rather, there are a number of parameters that may be varied:

Table 1: Dimensions of Web service features

| Dimension | Choices | |
| --- | --- | --- |
| | "Traditional" | Semantic Web |
| Service | Simple | Composed |
| Requestor | Human | Machine |
| Provider | Registration | No registration |
| Broker | Key Player | Facilitator |
| Service description | Taxonomy | Ontology |
| Descriptive elements | Closed world | Open world |
| Data exchange | Syntactic-based | Semantics-based |

In particular, the Semantic Web will allow to give richer descriptions of Web services (e.g., semi-structured data, types, inheritance, semantic constraints). The key role of the broker may disappear, it may still be viable as a kind of search machine for Web services (with meta search engines on top), but it will lose its central role, because everyone may publish semantic descriptions and crawlers may find them. Personalized machine agents will take over the role of a service requestor from the human user. And, they may also do the composition for the human user.

When we distinguish between "traditional" and Semantic Web features, we do, however, not require that only the latter be valid for two reasons. First, we do want to allow for an evolutionary developement of Web Service architecture. Second, also "traditional" features offer several advantages. For instance, a service broker as *intermediary* may take care of concerns like validation of information or even selection of service providers.

## Service Description with DAML-S

DAML-S is a DAML+OIL-based Web service ontology, which supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-intepretable form. DAML-S markup of Web services facilitates the automation of Web service tasks including automated Web service discovery, execution, composition and interoperation. In particular, it provides language primitives for technical, business-related and process-based facts about services. Thus, DAML-S can be regarded as a semantics-based substitution of the above-mentioned Web

service languages for service description, service publication, and service flow (cf. (Ankolenkar *et al.* 2001)):

- The *Service Profile* of DAML-S describes "what the service does" (business-related facts); that is, it provides a service-seeking agent with an abstract specification of a service to determine whether it meets the agent's needs (e.g. input and output types, preconditions and postconditions).

- The *Service Model* demonstrates "how the service works" (process-based facts); that is, it describes what happens when the service is performed. This description may be used by a service-seeking agent in at least four different ways: to perform a more in-depth analysis of whether the service meets its needs; to compose service descriptions from multiple services to perform a specific task; to coordinate the activities of different participants during the course of the service enactment; to monitor the execution of the service.

- The *Service Grounding* specifies the details of how an agent can access a service (technical facts). It will specify a communication protocol, e.g. Remote Procedure Call (RPC), Hyper-Text Transfer Protocol (HTTP), and Agent Communication Language (ACL), and service-specific details such as port numbers used in contacting the service.

Currently, only the first two have been described in some more detail.

## Critique of DAML-S

Though there are some points *within* DAML-S that may require discussion and possibly revision, we do not want to tackle them here. Rather, we want to consider the role that DAML-S plays with regard to *other* specifications like the IBM Web Service architecture:

- DAML-S' strength is the semantically-based definition of concepts. The corresponding languages like WSDL are rather weak in this respect.

- DAML-S is indecisive about how it should be realized in a particular architecture (though in several papers a shallow architecture is suggested in illustrations). This may be considered an advantage (many possibilities) or a disadvantage (increased complexity). We consider it a disadvantge for most practical purposes (no reliability on architectural issues).

- DAML-S does not include the role of an intermediary into its schemes (though it does not prohibit one either). Nevertheless, we consider the role of registries and service brokers as very important in order to increase quality of service, reliability, etc.

In the following, we describe the Semantic Web Service architecture — a proposal that aims to combine the advantages of the two worlds.

## Semantic Web Service Architecture

Ontologies provide a large extent of flexibility and expressiveness, the ability to express semi-structured data and con-

straints, and support types and inheritance. The industry's Web service (quasi-)standards, however, provide better manageability, scalability, and modularization. The benefits of both technologies can be obtained by merging DAML-S and current Web service standards to the benefits of both.

**Web Services and Web Service Access**

For this purpose, we distinguish two main scenarios. In the first one, a human user wants to access the Web service. There are (mainly) two possibilities: The user may directly access the Web service, (cf. arrow 1 in Figure 2), *viz.* the technology-based details of the service description, the business-related details of the service publication, and the internal process flow details of the service in service interaction. This direct invocation, however, is only possible if the user knows the service and its location, which often is not the case. Alternatively, the user may interact with a registry located at the service discovery layer which helps her finding a service and retrieving the descriptions about how to invoke it (arrow 2 in Figure 2).
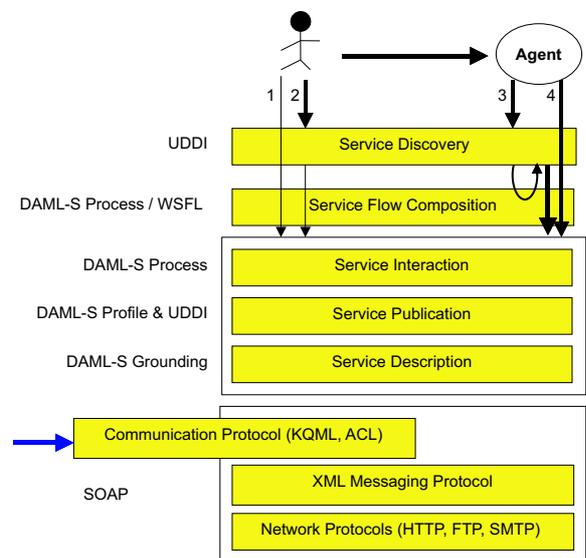


Figure 2: Semantic Web Service Architecture

In the second scenario, agents want to find and invoke a Web service in order to fulfill a task given to them by a human user. This adds new possibilities to the first scenario, because agents may better find services and access them directly (arrow 4 in Figure 2) exploiting the crawling of decentralized machine-processable metadata on the WWW. Furthermore, they may also take advantage of registries — just like a human user (arrow 3 in Figure 2).

In case, the human or the machine agents query for a composed service, e.g. one may ask for arrangements to visit a conference, the registry will make use of a service flow composition description. The description describes services and service flows (e.g., temporal sequences). With the help of this description, the services contained in the composed service (atomic or composed itself) may either be located directly or retrieved via the very same registry for atomic

services.

**Communication.** Like in the IBM quasi-standard, the two bottom layers are standard network protocols and SOAP in order to allow for exchange of object descriptions by standard means. In addition, the purpose of machine agents requires interaction with a corresponding communication protocol such as Knowledge Query and Manipulation Language (KQML) or Agent Communication Language (ACL) (cf., e.g. the survey paper (Labrou 2001)). These protocols are currently not "ontology-capable". Thus, there is a need for extension in order to better reflect ontology requirements at these levels.

**Further Layers.** Finally, the extended layers for Quality of Service, security, and management as depicted in the IBM Web service architecture are still needed, but are not illustrated in the figure for ease of presentation.

**The Role of DAML-S.** DAML-S is used for the full service description, *viz.* the technical, business-related, and process-based description of services. For the service publication layer, DAML-S should have a core ontology modeled with the UDDI properties and should offer further functionality with more properties. At the next layer, *viz.* service flow composition (corresponding to the service flow layer in the IBM Web service architecture), a language like WSFL or the DAML-S subontology `Process` should be in charge of describing the relations between services. The service discovery in the topmost layer will be performed by an ontology-compatible registry, which is capable to point to DAML-S descriptions and support their expressiveness. UDDI needs to be developed into this direction. In the long run, the currently mostly passive registry should be substituted by an active agent that functions as an intermediary. By this way the update of publication information does not only depend on active registration, but it may also be informed by the intermediary, the tasks of which may include the crawling of Web service descriptions.

**Web Service Binding.** The Semantic Web service architecture supports better service invocation by the use of agents because the underlying ontologies are extensible. This advantage is inherited by DAML-S, which allows for extension of the predefined ontologies.

## SWOBIS

We implemented the Semantic Web Ontology-Based Information Service (SWOBIS), which is analogous to the recommended UDDI registry which takes descriptions of services expressed as metadata to enable service discovery. It is a kind of ontology-compatible registry due to the following functionality: It offers a self-updating list of software tools for the Semantic Web generated as a Web-based report by utilizing the descriptions of software tools provided as metadata on the WWW. In addition to showing the next step in realizing the Semantic Web Service Architecture, SWOBIS keeps the research community updated about the performance of current Semantic Web technologies, thus supporting the vision of Semantic Web. The information about the latest development of software tools provides opportunities of collaboration and eliminates duplication of work. Further,

it is one of the first services to show the potential of existing Semantic Web tools and thus leads to motivation of both information and service providers to create DAML+OIL metadata. The advantages of SWOBIS can be seen in comparison to the existing list of software tools on the DAML project Web site at http://www.daml.org/tools/, which has shortcomings, such as having to communicate with the human maintainer to add to the list.

**Implementation of SWOBIS.** In the first step, the Semantic Web Software Tools (SWST) ontology was created, which is needed to describe software tools in a machine-understandable way. SWST models in a simplified way the Semantic Web software tool development activity: categorization of software tools, their features, the developers, the supporting organizations, etc. and relations between them. It is also possible to use another ontology to describe software tools and publish the metadata created according to it in the SWOBIS list. This can be realized by publishing ontological mappings. SWOBIS is capable of interpreting these mappings and thus can visualize information from various ontologies in one report.

In general, each researcher willing to provide metadata creates and makes it available on an annotated Web page.

For the presentation of SWOBIS we used WebScripter[5], a tool that enables ordinary users to produce easily and quickly live reports extracting and fusing information from multiple, heterogeneous DAMLized Web sources, thus taking account of the dynamics of the WWW.

To provide the self-updating list of software tools, the SWOBIS Web site at http://tools.semanticweb.org has been set up, a domain of the portal SemanticWeb.org.

**Results of SWOBIS.** SWOBIS offers various benefits in comparison to the current simple list: The report always contains up to date information. The user saves time that would have been otherwise spent retrieving information by looking at the list instead of visiting all Web pages of software tools that are of interest to him. The timesaving might be higher if personalization of the spreadsheet is available or agent interaction substitutes the human tasks. Due to the automatic update of information, more details about the software tools can be provided in the list. The classification of the software tools in the appropriate categories is performed by domain experts in the metadata creation process and thus should be more accurate. Transparency of the domain of interest attracts non-researchers, thus leads to an increase of audience and brand recognition of the tools. If personalization is accomplished, personalized reports will enhance user-friendliness.

The full potential of SWOBIS, however, relies in its use in the future Semantic Web service architecture. That is, SWOBIS needs to be described with metadata according to DAML-S and thus made available for agents to retrieve information about software tools.

## Related Work

This section reviews the architecture of other Semantic Web services (for humans, for machines, or for both).

---

[5]http://www.isi.edu/webscripter

ITTALKS is a web portal that lists information technology talks, such as distinguished lectures at universities. It internally uses DAML for knowledge representation, reasoning, and communication (Cost *et al.* 2001). A user can fill out a standard Web form that will result in a DAML file containing data that ITTALKS can process, which the users can then host on their own Web server. Talk information can similarly be entered via traditional HTML forms, or one can submit the URL of a DAML file using the standard ITTALKS content. Talks information is kept in a database much like in any other (non-Semantic-Web) Web service. ITTALKS does not provide for robust direct external access to its underlying DAML content (as of the time of writing, to the best of our knowledge). This means that it is impossible to robustly republish, search, or annotate its content – negating the key idea of the Semantic Web. Thus, while ITTALKS is a novel and immediately useful "traditional" Web service, its internal use of DAML seems to provide few benefits to external users.[6]

One division of USC's Information Sciences Institute automatically pulls together its people page from different DAMLized pages [7]. Some information is maintained by individual employees themselves (such as their research interests), other information is maintained by the division director (such as project assignments), and some information is maintained at the institute level (such as office assignments); this e.g. relieves administrative assistants from manually maintaining everyone's interests. The page also uses WebScripter as does SWOBIS; all the underlying DAML is available externally – scroll to the end of the cited Web page and click on "report".

The DAML Web site maintains a DAML tools[8] list in DAML format in the same spirit of SWOBIS; however, external users cannot add to the list without human intervention like they can do in SWOBIS. (It is a simple static Web page, not a service in any fashion.)

DAML-S is a promising add-on to the DAML+OIL language for describing services (Ankolenkar *et al.* 2001). In fact, the need for a merger between DAML-S and current industry quasi-standards has even been recognized by the developers of DAML-S. However, to the best of our knowledge, it has not yet been specified in any way nor has there been implemented Web services using DAML-S at the time of writing. Therefore, we withhold judgement on its viablity until a reference implementation exists.

UPML, the Unified Problem-solving Method Development Language, sits on top of the DAML+OIL layer just like DAML-S, and defines an architecture for describing reasoning services on the Web (Fensel *et al.* 1999). Using UPML, IBROW aims to develop intelligent brokers that are able to configure reusable components into knowledge systems via the World-Wide Web (and aims to do so in a distributed fash-

ion) (Benjamins *et al.* 1998). This is an ambitious project; various software pieces such as UPML editors exist, but there are no implemented distributed reasoning services to our knowledge, thus we similarly withhold judgement.

## Conclusion

We presented a Semantic Web Service architecture that integrates industry quasi-standards and academic research. We argued that such an integration is necessary in order to allow for a stepwise evolution of current standards towards the full benefits of Semantic Web technologies. However, we left many implementation details of the future full-featured Web services arachitecture open. Some of them were specified in (Sollazzo 2001), others are open because of the evolving status of the Semantic Web, e.g. the need for Semantic Web rule languages has been recognized, but not yet accounted for in the W3C documents.

## References

Ankolenkar, A.; Burstein, M.; Hobbs, J.; Lassila, O.; Martin, D.; McIlraith, S.; Narayanan, S.; Paolucci, M.; Payne, T.; Sycara, K.; and Zeng, H. 2001. DAML-S: A Semantic Markup Language For Web Services. In *Proceedings of SWWS'01, Stanford, USA, August 2001*.

Benjamins, V. R.; Plaza, E.; Motta, E.; Fensel, D.; Studer, R.; Wielinga, B.; Schreiber, G.; Zdrahal, Z.; and Decker, S. 1998. IBROW3: An intelligent brokering service for knowledge-component reuse on the world-wide web. In *The 11th Banff Knowledge Acquisition for nowledge-Based System Workshop (KAW98)*.

Cost, R. S.; Finin, T.; Joshi, A.; Peng, Y.; Nicholas, C.; Chen, H.; Kagal, L.; Perich, F.; Zou, Y.; and Tolia, S. 2001. Ittalks: A case study in the semantic web and daml. In *International Semantic Web Working Symposium*. Stanford University.

Fensel, D.; Benjamins, R.; Motta, E.; and Wielinga, B. 1999. UPML: A Framework for knowledge system reuse. In *Proceedings of the International Joint Conference on AI (IJCAI-99)*.

Horrocks, I.; Harmelen, F.; Patel-Schneider, P.; Berners-Lee, T.; Brickley, D.; Connolly, D.; Dean, M.; Decker, S.; Fensel, D.; Hayes, P.; Heflin, J.; Hendler, J.; Lassila, O.; McGuinness, D.; and Stein, L. 2001. DAML+OIL language release. http://www.daml.org/2001/03/daml+oil-index.html.

Kreger, H. 2001. Web Services Conceptual Architecture (WSCA 1.0). http://www-4.ibm.com/software/solutions/webservices/.

Labrou, Y. 2001. Standardizing Agent Communication. In *9th ECCAI Advanced Course ACAI 2001 and Agent Link's 3rd European Agent Systems Summer School, EASSS 2001*, volume 2086 of *Lecture Notes in Computer Science*. Prague, Czech Republic: Springer.

Leymann, F. 2001. Web Services Flow Language (WSFL 1.0). Technical Paper. http://www-4.ibm.com/software/ solutions/webservices/pdf/WSFL.pdf.

Martin, J. 2001. Web Services: The Next Big Thing. *XML-Journal* 2. http://www.sys-con.com/xml/.

Sollazzo, T. 2001. Ontolgoy-based Services for the Semantic Web. Services in the Area of Software Tools and Supply Chain Mangement. Master's thesis, University of Karlsruhe.

Trastour, D., and Bartolini, C. 2001. Approach to Service Description for Matchmaking and Negotiation of Services. In *Proceedings of SWWS'01, Stanford, USA, August 2001*.

---

[6]Note that SWOBIS provides direct external access to all of its underlying DAML input files – click on the "WebScripter report definition" link on any of the SWOBIS reports to retrieve the list of raw DAML input files.

[7]http://www.isi.edu/divisions/div2/people.html

[8]http://www.daml.org/tools/