

---

## Describing web services with semantic metadata

---

### Sinuhé Arroyo

DERI-Innsbruck,  
Technikerstrasse, 13 A-6020, Innsbruck, Austria  
E-mail: sinuhe.arroyo@deri.org  
Website: <http://Deri.Semanticweb.Org>

### José Manuel López-Cobo\*

Atos Origin S.A.E., Albasanz 12 E-28037 Madrid  
E-mail: jose.lopez@atosorigin.com  
\*Corresponding author

**Abstract:** Web services are self-describing, self-contained applications that can be published, located, and invoked through common web protocols. Self-descriptions are in fact a form of metadata that provide details on the services offered. Semantic web services are, thus, an extension of such metadata-based descriptions to richer ontology-based description semantics. This paper provides an account on the main usage scenarios of semantic web services, as a roadmap for metadata research on the topic.

**Keywords:** metadata; semantics; ontology; web services; semantic web services.

**Reference** to this paper should be made as follows: Arroyo, S. and López-Cobo, J.M. (2006) 'Describing web services with semantic metadata', *Int. J. Metadata, Semantics and Ontologies*, Vol. 1, No. 1, pp.76–82.

**Biographical notes:** Sinuhé Arroyo is a PhD researcher at DERI in the area of Semantic Web Services. He obtained his MS in Computer Science at Universidad de Málaga, Spain in 2000 and his BS in Software Engineering at University Complutense de Madrid, Spain in 1997. Sinuhé has published numerous peer-reviewed papers in international conferences and has an extensive teaching experience. While in DERI he has been project manager of the EU-funded projects Esperanto and DIP, and contributed to the elaboration of numerous funded proposals under the EU-framework 6 program. Previously he was Project Leader at iSOCO in Madrid, Spain.

José-Manuel López-Cobo is a PhD researcher at the University of Alcalá de Henares in the area of Semantic Web Services and eLearning. Besides that he is also working in Atos Origin as technical manager of the Software and Services Unit, developing research projects. He obtained his MS in Computer Science at Universidad de Málaga, Spain in 1998. José-Manuel has focused his career in the management and research in EU funded projects, gathering some expertise in the field of the Semantic Web (iBROW), the Semantic Web Services (SWWS, DIP, INFRAWESBS) and in the Service Centric approach (SeCSE).

---

## 1 Introduction

Web services are self-contained, self-describing software applications that can be published, discovered, located, and invoked across the web, using standard web protocols. Nowadays, the basic support for web services is provided through the SOAP, WSDL, and UDDI specifications, which address message format, service description, and service publishing and lookup, respectively.

Nonetheless, this basic support still provides limited help in automating configuration and combination, which has fostered proposals like DAML-S and WSMF that employ semantic web technologies for service description

and related aspects. Using metadata to annotate functional behaviours like Service Descriptions allow us to ease the automation of bring together different services and execute them to obtain a more complex result.

This paper attempts to provide a depiction of the *functions* (Greenberg, 2003) that should be supported by service metadata, providing a roadmap for research on the diverse aspects that are required in full-fledged semantic web service architecture. Section 2 provides an introduction to semantic web services, and the main functions that are to be supported are described in Section 3. Final conclusions are provided in Section 4.

## 2 Semantic web services

Current state-of-the-art technologies around web services present a gap in stating what services can do, and how they do it in a machine-understandable way. In order to bridge such gap and fully develop the web and web services, it requires the addition of meta-data. Such metadata provides the proper support for machine understandability and processability; thus, making the web a more dynamic place, where resources can be published, located, and used as required. Human intervention will as a consequence be reduced, transforming the web into a more efficient place, where business transactions are more agile, and the integration of business logics can be done in a task-driven way.

The combination of machine processable semantics provided by the semantic web, with current web service technologies has coined the term semantic web services. Semantic web services represent the dynamic part of this emerging web. They offer the means to achieve a higher level of value-added services by adding dynamism to the task driven assembly of inter-organisation business logics, thus, making the internet a global and common platform, where agents (organisations, individuals, and software) communicate with each other to carry out various activities, providing new value-added services.

They go a step further in extending the web from a distributed source of information to a distributed source of services (Lara et al., 2003), by providing a higher degree of dynamism and automation in terms of machine processability and machine-human understanding; thus, being the next natural step in the development of web service technology.

Ontology facilitates the machine processable semantics that along with current web services realise the idea of the semantic web services. Semantic web services are defined as

“Decoupled, semantically marked-up web services (Tidwell, 2000; W3C, 2002), with concrete execution semantics, that can be published, discovered, selected, composed, mediated and executed across the web, in a task driven way, carrying its interaction (Arroyo et al., 2004c) following a choreographed or orchestrated approach.”

The definition states different desired aspects of web services that require further clarification:

- *Decoupling*. The internal business intelligence of applications should be hidden from public access, differentiating between the public (interface) and private (implementation) part of services. The public part of a service should use messages' exchange protocols to communicate, thus enabling disintegration as much as possible.
- *Semantically marked-up*. Semantic web services are described in terms of its capability and interface (Roman et al., 2004). Both, capability and interface need to be semantically marked-up, in order to allow its

dynamic use and interoperation. These metadata are specified in a formal language in order to allow a fully machine processability.

- *Execution semantics*. Interactions among services require formal execution semantics, in order to uniquely specify their execution behaviour.
- *Mediation*. In a scalable environment, communications should allow anybody to speak with everybody. Mediation is the pivotal element that allows the use of different communication patterns, business logics, and data exchanged among interacting parties.
- *Choreography vs. orchestration*. When cooperating services communicate in order to achieve some functionality, the interaction can be classified into two different categories, choreography and orchestration, depending on the level of control they have over the other party. While in choreography, one party controls the interaction process defining how to communicate with the service in order to consume its functionality, in an orchestrated interaction, the level of control of all the parties is the same, thereby, achieving the overall functionality by the cooperation of the different services.

Semantic web services, the same as the semantic web itself, represent an extension to current web services technology (Arroyo et al., 2005). The main difference with respect to traditional web services is that semantic web services are semantically enhanced with metadata. Such enhancement allows further realising the concepts behind the semantic web services Usage Process, namely: publication, discovery, selection, composition, mediation, execution, monitoring, compensation, replacement, and auditing, at least in a more dynamic and semi-automatic way.

Nowadays, technology presents a gap between what the current web offers and the requirements of semantic web services. The semantic web in general and ontology in particular are the right means to bridge the gap, and realise a dynamic web. They provide the machine processable semantics that added to the current web services realise the idea of the semantic web services. In order to address the open research issues, it is important first to enlist the problems that form part of the semantic web service framework. These problems are described in the following section by logically following the usage process of services.

## 3 Semantic web services usage process

The semantic mark-up of services allows the description of its capabilities in terms of the functionality that they provide and the interfaces allowing the means to understand how to use their functionality for the benefit of some agent, who seeks to discover them, determine how to execute them, and additionally may want to combine them with other ones in order to produce some aggregated functionality (Arroyo et al., 2004a).

The process of publishing, discovering, and executing services carried with the aim of fulfilling some user-defined task, conceptualised as a goal, has been termed semantic web service usage process (Arroyo and Stollberg, 2004). The aims of the service usage domain are wide, not being limited to just publishing, discovery and execution. These core steps are complemented with selection, composition, mediation, monitoring, replacement, compensation, and auditing, thus, covering all the different aspects involved. The ultimate goal of such process is allowing to:

- make available the description of the capability and interface of a service (publication)
- locate different services suitable for a given goal (discovery)
- select the most appropriate services among the available ones (selection)
- combine services to achieve the goal (composition)
- solve mismatches (data, protocol, process) among the combined services (mediation)
- invoke services following programmatic conventions (execution)
- control the execution process (monitoring)
- facilitate the substitution of services by equivalent ones (replacement)
- provide transactional support and undo or mitigate unwanted effects (compensation)
- verify that service execution occurred in the expected way (auditing).

The following sub-sections detail the main of these goals.

### 3.1 Overall behaviour

In order that successful discovery occurs, the publisher needs to facilitate a description of the relevant information of the service/s to be made available in the repository. As far as discovery is concerned, this refers to the capability of the service. Eventually, an end user will specify a goal, which must be translated into a machine-readable discovery query. Such a goal is decomposed into constituent's sub-goals that are matched against the capabilities of the registered services, choosing the ones that by themselves or in combination with others allow achieving the objective. During the discovery phase, mediation is required to allow services described, using different domain knowledge to be matched against the goal and vice-versa. Once the list of matching services has been retrieved, a selection process is to be carried, based on the functional and non-functional properties of the service, with the aim of picking the most suitable ones, e.g., the most reliable and cost effective.<sup>1</sup>

The list of preferred services is then made available to the composition stage. In case composition is required – one service might be sufficient by itself to acquire the goal – the list of services is assembled in a way that enables their

interoperation. Mediation is required to overcome mismatches and further to allow interoperation, with regard to data formats used in the message exchange, protocols used to interact, and specific business models.

Finally, the service is to be executed at the expenses of the service provider. In the case of composed services, execution requires additional support for monitoring, compensation, replacement, and auditing. While a service is being executed, different aspects could lead to failure. Monitoring is concerned with determining the current state of a service execution, possibly checking the progress of an order, or finding out what state a service is at in the event of a reported problem (Duke et al., 2004). In case a problem occurs, the compensation support will take care of repairing or undoing the effects of the execution. In order to continue with the normal execution of the service, a new one or combination of new ones could be required. Replacement support cares for this particular aspect. To do so, it will eventually start the usage process from the beginning, trying to discover services whose capability match the one of the malfunctioning service. Auditing represents the last step of the execution of a service allowing the determining of whether or not services have run as expected.

### 3.2 Publication

Service publication can be defined as: “The process of making the relevant information of a semantic web service available to others, in order to allow its further use”.

Service publishers are in charge of describing the main aspects of services, and making that information available, so that it can be discovered. Different aspects of services need to be described. Among the most relevant ones are counted (Roman et al., 2004):

- *Capability*. Description of a web service by means of its functionality, based on pre-conditions, post-conditions, assumptions, and effects.
- *Goal*. Specifies the objectives that a client may have when he consults a web service based on post-conditions and effects. A goal can be made of constituent sub-goals.
- *Interface*. Specification of how the functionality of the service can be achieved, by means of fulfilling its capability. It takes a two-fold approach, based on choreography, or how to communicate with the web service in order to consume its functionality, and orchestration, which defines how the overall functionality is achieved by the cooperation of more elementary service providers.
- *Non-functional properties*. Properties related to quality aspects of web service (QoS).

The semantically marked-up descriptions of services are to be registered in semantically enhanced service repositories, where they can be easily located and their capabilities matched against the user's goals, the service requester being the responsible agent for conducting the discovery phase.

### 3.3 Discovery

Service discovery can be defined as: “Location of services whose capability satisfies the service requester specification of a goal” (Arroyo et al., 2004b).

In a nutshell, discovery can be understood as the matchmaking task of the capabilities of services (pre-conditions and assumptions) against the goal (post-conditions and effects) that the end user aims to achieve. The idea is to locate a selection of services, which by themselves or in combination with others allow accomplishment of a particular goal. For this, the user goal is decomposed in sub-goals that are submitted as discovery queries.

Matching requires reasoning support, both for goal capability resolution, as well as domain knowledge mediation, that it is not provided as part of the registry functionality (Arroyo et al., 2004e).

### 3.4 Selection

Service selection can be defined as: “Choice among discovered Services, based on functional and non-functional attributes, of those that better suit the user requirements”.

The discovery phase returns the set of candidate services whose functional characteristics can fulfil the goal. It is the task of the service requester, to choose the most appropriate ones, based on their non-functional properties. A common feature of the services that reach the selection phase is that they can be used to at least, to partially solve the goal – solution to a sub-goal –. In case no appropriate services are found, or the ones found are not able to fully realise the goal, the service requester should inform the end user. This should either refine the description of the discovery query in case the partial solutions are not sufficient, or proceed with a group of services.

The number of retrieved services might be really vast, yielding the necessity for the service requester to browse over an enormous amount of service descriptions. Various techniques for automating the selection of services have been explored (Li and Horrocks, 2003; Paolucci et al., 2002). Usually, some ranking algorithm is employed to order the services picking automatically the first ranked ones suitable for the goal.

### 3.5 Composition

Service composition can be defined as: “Assembly of services based on their functional specifications in order to achieve a given task and to provide a higher order of functionality” (Arroyo et al., 2004b).

From a component-based software engineering (CBSE) (Arroyo et al., 2004d) point of view, web services can be envisioned as reusable pieces of software that are independently assembled, deployed, and plugged in any context that requires its functionality further enabling a new level of service, by means of making accessible more complex functionalities that in turn can be used to compose new services. Under this assumption, web service

composition becomes a matter of reuse. Roughly speaking, it specifies how to assemble pieces of software – components – that represent the basic building blocks of a system, independently developed and delivered from various domains and/or vendors.

Once a list of preferred web services has been defined, it could happen that none of the available services completely fulfils the proposed task alone, or other – cheaper, faster, vendor-dependent – combinations of services are preferred. If this were the case, some of the services would need to be organised in different possible ways, based on their capabilities, using programmatic conventions to accomplish the desired goal.

Mediation support is of major importance in this part of the process. Since services are arranged based on its functional characteristics to allow interoperability, all the different mismatches that could occur among them should be overcome in this phase. Such mismatches affect the data, protocol, and process level description of the service.

As an example, the end user might say, “Compose available services to solve the following set of mathematical operations  $((a \times b) + c) - d$ ”. During the discovery phase, different multiplication, addition, and subtraction services might have been found, each one of them with its particular functional and non-functional attributes. Let us suppose that some multiplication services have been located, but they are all too slow and expensive, so we are not interested in using them. Instead we want to use additions to perform the multiplication. Such knowledge, i.e., the fact that multiple-addition can realise multiplication, should be stated in some domain knowledge in a way that the service composer can understand all different possibilities to solve the required set of operations.

### 3.6 Mediation

Service mediation can be defined as:

“Arbitration of interacting Services in terms of domain knowledge used to describe them, data exchanged in the interaction (types used, and meaning of the information), protocol used in the communication and business models of the different parties (Arroyo and Fensel, 2004).”

Mediation is the pivotal element of the usage process. It is very inter-leaved in all the different steps of the process and relies on the use of ontology to carry out its task. In a typical web service scenario, the requester sends messages to the provider using his own vocabulary and protocol of choice within its own business model, who needs to interpret messages received in terms of his own vocabulary protocol and business model.

Depending of how the interfaces of the interacting parties are defined (choreography or orchestration), problems arise when communicating with a number of providers, in which case the messages need to be translated specifically for each provider. Ideally, a mediator resolves the differences in representation between the requester and the provider.

Once all the required services to solve a task are brought together following some programmatic conventions, interacting services need to be mediated in terms of:

- *Types and meaning of the exchanged information : data mediation.* Depending on the application domain for which the services are deployed, different data types and domain knowledge might be used to encapsulate data and its meaning, which requires mediation to allow interoperation
- *Protocol used: protocol mediation.* Different parties might use different message exchange patterns and protocols that need to be mediated in order enable communication.
- *Business models: process mediation.* Services belonging to different business models require the appropriate process mediation in order to permit their cooperation.

With regard to the relation among the different types of mediation, it interesting to note that process mediation, in a broad sense, relies in data and protocol mediation. When taking a bottom-up approach, first all data types and their meaning are mediated, then the appropriate protocol mediation, if required is carried, and finally process mediation can be done.

### 3.7 Execution

Service execution can be defined as: “Invocation of a concrete set of services, arranged in a particular way following programmatic conventions that realizes a given task” (Arroyo et al., 2004d).

When the available services have been composed and the service requester has chosen the execution path that best suits its needs, according to the functional and non-functional requirements, the chosen set of web services is to be executed.

Service execution has to deal with the dynamic invocation of services. Frameworks such as CORBA have already addressed the problem in the field of distributed object oriented programming. Essentially the service requester needs to figure out how to invoke a service without having a static proxy. Introspection and brokers are techniques used in object-oriented programming to do this. With regard to web services, the grounding mechanism allows the service requester to go from the conceptual to the concrete description of that service.

Each web service specifies one or more signatures of its operations that allow its execution, defining the inputs required and outputs returned, describing the format to put messages in, figuring out how to physically send a message to the service, and being able to follow a specified protocol for interacting with the service. The semantic mark-up of services makes all this information available thereby realising service execution in more dynamic way.

Prior to its execution, the service may impose some limitations on the service requester. Such restrictions are

expressed in terms of assumptions – conditions about the state of the world. A service provider may state that the service requester has to have a certain amount of money in the bank prior to the execution of the service, or any other requirements considered necessary as part of a concrete business model. In addition, the service execution requires the facilities offered by composition engines to coordinate and control the whole process, particularly in the case of complex process logic of composite service.

### 3.8 Execution support

Along with the core steps of publication, selection, discovery, composition, execution, and mediation the execution of services requires many other important aspects that need to be cared for. Under the term execution support can be found aspects such as monitoring, compensation, replacement, and auditing.

Monitoring is concerned with detecting the malfunctioning of any of the services currently being executed further determining their current state. In the event of failure, the compensation mechanism is responsible for repairing or undoing the effects of the execution, while replacement takes care or putting into place a new service or combination of services to carry on with execution. Finally auditing allows the determining of whether or not services have run as expected.

The remaining of this section carefully elaborates on these aspects, providing a detailed overview of the most relevant characteristics.

#### 3.8.1 Monitoring

Service monitoring can be defined as: “Real-time supervision of the correct execution of services, determining the current state of execution and dealing with exceptions thrown by composed services or the composition workflow itself” (Arroyo and Fensel, 2004).

Monitoring requires the internal operation of services to be described for the purpose of allowing the service requester or final user to control the status of the execution. It facilitates the appropriate means to find out what the state of a service is at the event of a reported problem, providing appropriate support to deal with the exceptions thrown by the services or the composition workflow.

The main reasons that could cause a failure in the execution of a service are categorised and described as follows:

- *service related issues:* the server hosting the service crashes, or the service throws an exception that does not allow continuing execution
- *network related issues:* network crashes disallowing communication
- *composition related issues:* the composition workflow throws an exception that does not allow continuing execution, or it is not well designed so it reaches a dead end.

On the one hand, monitoring should facilitate the means to supervise the individual aspects of services taking part on the composition. Due to the distributed and decoupled nature of services and the amount of data that it needs to deal with, monitoring is not a trivial task at all, requiring that the execution environment, by means of its central monitoring component collect information from the particular services. It should also provide support to visualise input and output parameters, timestamps and log facilities for the different interactions, carrying performance measures.

### 3.8.2 Compensation

Service compensation can be defined as: “Reparation or undoing of the operations involved in a transaction in case of failure, so unwanted side effects are rolled back or at least mitigated if a complete undo is not possible”.

When the execution of services must follow a transactional model, either it is fully completed or not at all completed. There might be situations where the whole transaction cannot be fully finished. If this is the case, some of the effects of the previously executed services might not be undoable, requiring the application of compensation mechanisms in order to mitigate the unwanted effects, usually by executing another operation(s), whose effect compensate or make-up for the side effects. In the case the effects are undoable the appropriate rollback mechanism is performed, thereby undoing the completed parts of the transaction.

Compensating actions are considered part of the business logic of services, requiring their specification for each one of the operations carried (Duke et al., 2004).

In a nutshell, compensation strategies can be classified in two main categories, computational or software compensations and physical compensations. In the case of computational or software compensations, the effects of an action, i.e., insertion of information in a database or charging a bank account with a concrete amount, can be easily undone by another computational action. Physical compensations are normally derived from computational compensations, such as the cancellation of an order, possibly requiring the storage of the goods produced or the cancellation of some shipment.

### 3.8.3 Replacement

Service replacement can be defined as: “Substitution of services by equivalent ones, which solely or in combination can realize the same functionality as the replaced one, in case of failure while execution”.

In case any of the constituents of the execution workflow fails to accomplish its goal, e.g., network breaks down, service provider’s server breaks down, etc., a recovery procedure must be applied to replace the failed service with another one or set of services capable of finishing the work. To do so, it will eventually start the usage process from the beginning trying to discover services whose capability match the one of the malfunctioning one,

selecting the most appropriate ones, composing them if required and finally executing them.

The concepts of compensation and replacement are orthogonal, i.e., in case the execution of a service fails; there might be a need for compensation, in case the service produces side effects. Also, if a service fails, after compensating its side effects, we normally consider the replacement of this service. But such a link between the two operations is not necessary, i.e., there can be replacement without compensation (e.g., we are sure that there were no side effects) or compensation without replacement (e.g., we abort the whole process, if one service fails, and do not seek replacement).

### 3.8.4 Auditing

Service auditing can be defined as: “Off-line evaluation of the execution of services, composed services or the composition workflows itself”.

There exists a close relation between monitoring and auditing. The main difference is that monitoring occurs at run-time while auditing takes place as an offline activity, long after services execution took place (Duke et al., 2004). Also, while for auditing the collection and storage of execution data is crucial, in the case of monitoring services, they can be executed even if no data is gathered, thus improving performance (Duke et al., 2004).

Logs are used to store relevant data about the operations performed, being filled during service execution. Later on, and as a batch activity, they are processed and analysed to facilitate security maintenance, recovery from lost transactions, or to be compliant with legal affairs.

## 4 Current projects and research lines

Many initiatives are currently developing semantic web services and this topic has become one of the most important in both academic research and industrial applications. Some of the topics mentioned in this paper are being currently developed by the leaders of the semantic web and the business process management world. Two important initiatives, initiated in USA and Europe, respectively are leading the way to obtain results in the semantic web services. These two initiatives are OWL-S (OWL for Services, 2003) (formerly known as DAML-S from the DARPA program) and WSMO. The main lines in research, nowadays, are related to the composition of semantic web services, the establishment of a semantic environment of execution and the reasoning needed for the automated discovery of Services.

This paper has provided an overview of the ways by which metadata can ease the automation of the processes of publishing, discovery, selection, mediation, composition, and execution of semantic web services. Ontology can be used to describe not only the domain knowledge related to the services but also the process of interaction with services, with the possibility of combining them to fulfil a more complex task by means of the participation of several services.

A complete lifecycle of a semantic web service has been presented, explaining the concept and processes of the semantic web services usage process. Within this lifecycle, processes such as publishing, selection, discovery, mediation, composition, execution, compensation, auditing, monitoring, and replacement have been described.

## References

- Arroyo, S. and Stollberg, M. (Eds.) (2004) *WSMO Primer, WSMO Working Draft v0.1*, <http://www.wsmo.org/2004/d31/v01/>.
- Arroyo, S., Cimpian, E., Dimitrov, M., Domingue, J., Nagypal, G., Spork, M. and Vasiliu L. (2004a) *D3.2. Service Description Framework*, DIP project deliverable FP6 – 507483.
- Arroyo, S., Cimpian, E., Domingue, J., Feier, C., Fensel, D., König-Ries, D., Lausen, H., Polleres, A., and Stollberg, M. (2005) *Web Service Modeling Ontology Primer*, W3C Member Submission.
- Arroyo, S., Ding, Y., Lara, R., Stollberg, M. and Fensel, D. (2004b) ‘Semantic web languages. strengths and weakness’, *International Conference in Applied computing (IADIS04)*, Lisbon.
- Arroyo, S., Lara, R., Gómez, J.M., Berka, D., Ding, Y. and Fensel, D. (2004c) ‘Semantic aspects of web services’, in Singh, M.P. (Ed.): *Practical Handbook of Internet Computing*, Chapman Hall and CRC Press, Baton Rouge.
- Arroyo, S., Sung-Kook, H. and Fensel, D. (2004e) Searching for Semantic Web Services. A Google Based Approach, *Second International Conference on Information Technology (ICIT05)*, Amman, Jordan, 3–5 May 2005.
- Arroyo, S., Toma, I., Roman, D., Drumm, C., Dimitrov, M., Spork, M., Nagypal, G., Domingue, J., and Henke, J. (2004d) *D3.1 Report on State of the Art and Requirements Analysis*, DIP project deliverable FP6 – 507483.
- Duke, A., Richardson, M., Zugmann, P., Herzog, R., Quantz, J., Vayssiere, J. and Henocque L. (2004) *D41. Requirements and State of the Art Analysis for Service Usage*, DIP project deliverable FP6 – 507483.
- Greenberg, J. (2003) ‘Metadata and the world-wide-web’, *Encyclopaedia of Library and Information Science*, pp.1876–1888.
- Lara, R., Lausen, H., Arroyo, S., de Bruijn, J. and Fensel, D. (2003) ‘Semantic web services: description requirements and current technologies’, *International Workshop on Electronic Commerce, Agents, and Semantic Web Services, In Conjunction with the Fifth International Conference on Electronic Commerce (ICEC 2003)*, Pittsburgh, PA.
- Li, L. and Horrocks, I. (2003) ‘A software framework for matchmaking based on semantic web technology’, *Proc. of the Twelfth International World Wide Web Conference, WWW, ACM*, pp.331–339.
- OWL for Services (2003) <http://www.daml.org/services/owl-s/1.0/>.
- Paolucci, M., Kawamura, T., Payne, T. and Sycara. K. (2002) ‘Semantic matching of web services capabilities’, *Proc. of the 1st International Semantic Web Conference (ISWC)*.
- Roman, D., Lausen, H. and Keller, U. (Eds.) (2004) *Web Service Modelling Ontology, WSMO Working Draft v0.3*, <http://www.wsmo.org/2004/d2/v1.0/>.
- Tidwell, D. (2000) *Web Services: The Web’s Next Revolution*, <http://www.ibm.com/developerWorks>.
- Web Services Architecture Requirements W3C (2002) <http://www.w3.org/TR/wsa-reqs/>, October.

## Note

<sup>1</sup>A list of relevant non-functional properties can be found in Roman et al. (2004).